Arria V Avalon-MM Interface for PCIe Solutions

User Guide

Last updated for Altera Complete Design Suite: 14.1

UG-01105_avmm

2014.12.15

Subscribe

Send Feedback

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Datasheet 1

2014.12.15

UG-01105_avmm | Subscribe | Send Feedback

Avalon-MM Interface for PCIe Datasheet

Altera® Arria® V FPGAs include a configurable, hardened protocol stack for PCI Express® that is compliant with *PCI Express Base Specification 2.1 or 3.0*.

The Hard IP for PCI Express PCIe IP core using the Avalon ® Memory-Mapped (Avalon-MM) interface removes some of the complexities associated with the PCIe protocol. For example, it handles all of the Transaction Layer Protocol (TLP) encoding and decoding. Consequently, you can complete your design more quickly. The Avalon-MM interface is implemented as a bridge in FPGA soft logic. It is available in Qsys. The following figure shows the high-level modules and connecting interfaces for this variant.

Figure 1-1: Arria V PCIe Variant with Avalon-MM Interface

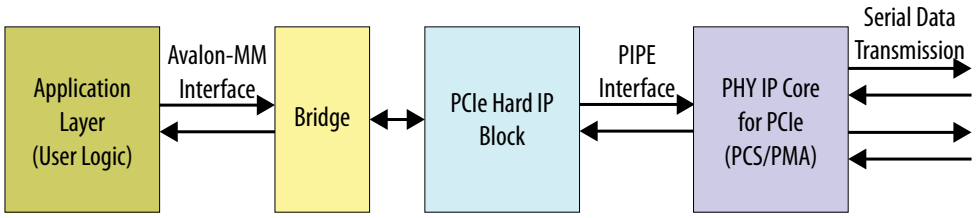


Table 1-1: PCI Express Data Throughput

The following table shows the aggregate bandwidth of a PCI Express link for Gen1 and Gen2 for 1, 2, 4, and 8 lanes. The protocol specifies 2.5 giga-transfers per second for Gen1 and 5 giga-transfers per second for Gen2. This table provides bandwidths for a single transmit (TX) or receive (RX) channel. The numbers double for duplex operation. Gen1 and Gen2 use 8B/10B encoding which introduces a 20% overhead.

	Link Width in Gigabits Per Second (Gbps)			
	×1	×2	×4	×8
PCI Express Gen1 (2.5 Gbps)	2	4	8	16
PCI Express Gen2 (5.0 Gbps)	4	8	16	N/A

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Refer to the *PCI Express High Performance Reference Design* for more information about calculating bandwidth for the hard IP implementation of PCI Express in many Altera FPGAs.

Related Information

- **PCI Express Base Specification 2.1 or 3.0**
- **PCI Express High Performance Reference Design**
- **Creating a System with Qsys**

Features

New features in the Quartus® II 14.1 software release:

- Reduced Quartus II compilation warnings by 50%.

The Arria V Hard IP for PCI Express with the Avalon-MM interface supports the following features:

- Complete protocol stack including the Transaction, Data Link, and Physical Layers implemented as hard IP.
- Support for ×1, ×2, ×4, and ×8 configurations with Gen1 and Gen2 lane rates for Root Ports and Endpoints.
- Dedicated 16 KByte receive buffer.
- Optional hard reset controller for Gen2.
- Optional support for Configuration via Protocol (CvP) using the PCIe link allowing the I/O and core bitstreams to be stored separately.
- Qsys example designs demonstrating parameterization, design modules, and connectivity.
- Extended credit allocation settings to better optimize the RX buffer space based on application type.
- Optional end-to-end cyclic redundancy code (ECRC) generation and checking and advanced error reporting (AER) for high reliability applications.

  Easy to use:

  - Flexible configuration.
  - No license requirement.
  - Example designs to get started.

Table 1-2: Feature Comparison for all Hard IP for PCI Express IP Cores

The table compares the features of the four Hard IP for PCI Express IP Cores.

Feature	Avalon-ST Interface	Avalon-MM Interface	Avalon-MM DMA
IP Core License	Free	Free	Free
Native Endpoint	Supported	Supported	Supported
Legacy Endpoint (1)	Supported	Not Supported	Not Supported

(1) Not recommended for new designs.

Feature	Avalon-ST Interface	Avalon-MM Interface	Avalon-MM DMA
Root port	Supported	Supported	Not Supported
Gen1	×1, ×2, ×4, ×8	×1, ×2, ×4, ×8	x8
Gen2	×1, ×2, ×4	×1, ×2, ×4	×4
64-bit Application Layer interface	Supported	Supported	Not supported
128-bit Application Layer interface	Supported	Supported	Supported
Transaction Layer Packet type (TLP)	• Memory Read Request • Memory Read Request-Locked • Memory Write Request • I/O Read Request • I/O Write Request • Configuration Read Request (Root Port) • Configuration Write Request (Root Port) • Message Request • Message Request with Data Payload • Completion Message • Completion with Data • Completion for Locked Read without Data	• Memory Read Request • Memory Write Request • I/O Read Request—Root Port only • I/O Write Request—Root Port only • Configuration Read Request (Root Port) • Configuration Write Request (Root Port) • Completion Message • Completion with Data • Memory Read Request (single dword) • Memory Write Request (single dword)	• Memory Read Request • Memory Write Request • Completion Message • Completion with Data
Payload size	128–512 bytes	128 or 256 bytes	128 or 256 bytes
Number of tags supported for non-posted requests	32 or 64	16	16
62.5 MHz clock	Supported	Supported	Not Supported
Multi-function	Supports up to 8 functions	Supports single function only	Supports single function only

Feature	Avalon-ST Interface	Avalon-MM Interface	Avalon-MM DMA
Out-of-order completions (transparent to the Application Layer)	Not supported	Supported	Supported
Requests that cross 4 KByte address boundary (transparent to the Application Layer)	Not supported	Supported	Supported
Polarity Inversion of PIPE interface signals	Supported	Supported	Supported
ECRC forwarding on RX and TX	Supported	Not supported	Not supported
Number of MSI requests	1, 2, 4, 8, or 16	1, 2, 4, 8, or 16	1, 2, 4, 8, or 16
MSI-X	Supported	Supported	Supported
Legacy interrupts	Supported	Supported	Supported
Expansion ROM	Supported	Not supported	Not supported

The purpose of the *Arria VAvalon-MM Interface for PCIe Solutions User Guide* is to explain how to use this IP core and not to explain the PCI Express protocol. Although there is inevitable overlap between these two purposes, this document should be used in conjunction with an understanding of the *PCI Express Base Specification*.

Note: This release provides separate user guides for the different variants. The *Related Information* provides links to all versions.

Related Information

- **V-Series Avalon-MM DMA Interface for PCIe Solutions User Guide**
- **Arria V Avalon-MM Interface for PCIe Solutions User Guide**
- **Arria V Avalon-ST Interface for PCIe Solutions User Guide**

Release Information

Table 1-3: Hard IP for PCI Express Release Information

Item	Description
Version	14.1
Release Date	December 2014
Ordering Codes	No ordering code is required
Product IDs	There are no encrypted files for the Arria V Hard IP for PCI Express. The Product ID and Vendor ID are not required because this IP core does not require a license.
Vendor ID	

Device Family Support

Table 1-4: Device Family Support

Device Family	Support
Arria V	Final. The IP core is verified with final timing models. The IP core meets all functional and timing requirements for the device family and can be used in production designs.
Other device families	Refer to the *Related Information* below for other device families:

Related Information

- **Arria V GZ Avalon-MM Interface for PCIe Solutions User Guide**
- **Arria V GZ Avalon-ST Interface for PCIe Solutions User Guide**
- **Arria 10 Avalon-MM Interface for PCIe Solutions User Guide**
- **Arria 10 Avalon-MM DMA Interface for PCIe Solutions User Guide**
- **Arria 10 Avalon-ST Interface for PCIe Solutions User Guide**
- **Cyclone V Avalon-MM Interface for PCIe Solutions User Guide**
- **Cyclone V Avalon-ST Interface for PCIe Solutions User Guide**
- **IP Compiler for PCI Express User Guide**
- **Stratix V Avalon-MM Interface for PCIe Solutions User Guide**
- **Stratix V Avalon-ST Interface for PCIe Solutions User Guide**
- **Stratix V Avalon-ST Interface with SR-IOV for PCIe Solutions User Guide**

Configurations

The Avalon-MM Arria V Hard IP for PCI Express includes a full hard IP implementation of the PCI Express stack comprising the following layers:

- Physical (PHY), including:
  - Physical Media Attachment (PMA)
  - Physical Coding Sublayer (PCS)
- Media Access Control (MAC)
- Data Link Layer (DL)
- Transaction Layer (TL)

When configured as an Endpoint, the Arria V Hard IP for PCI Express using the Avalon-MM supports memory read and write requests and completions with or without data.

Figure 1-2: PCI Express Application with a Single Root Port and Endpoint

The following figure shows a PCI Express link between two Arria V FPGAs. One is configured as a Root Port and the other as an Endpoint.

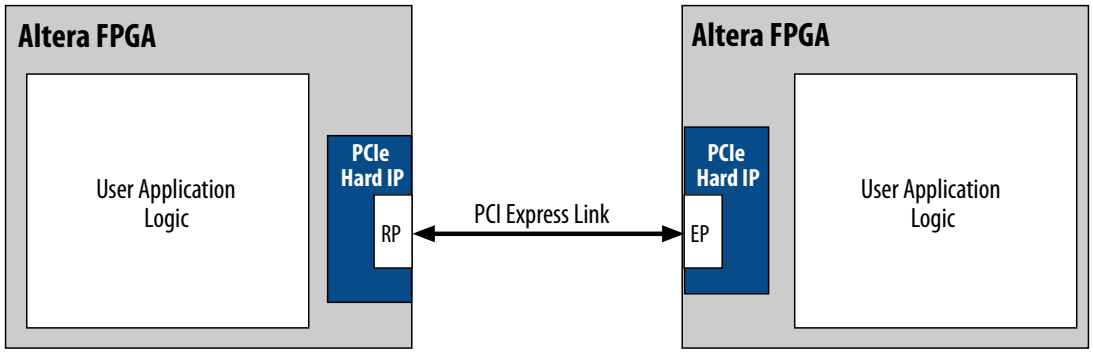
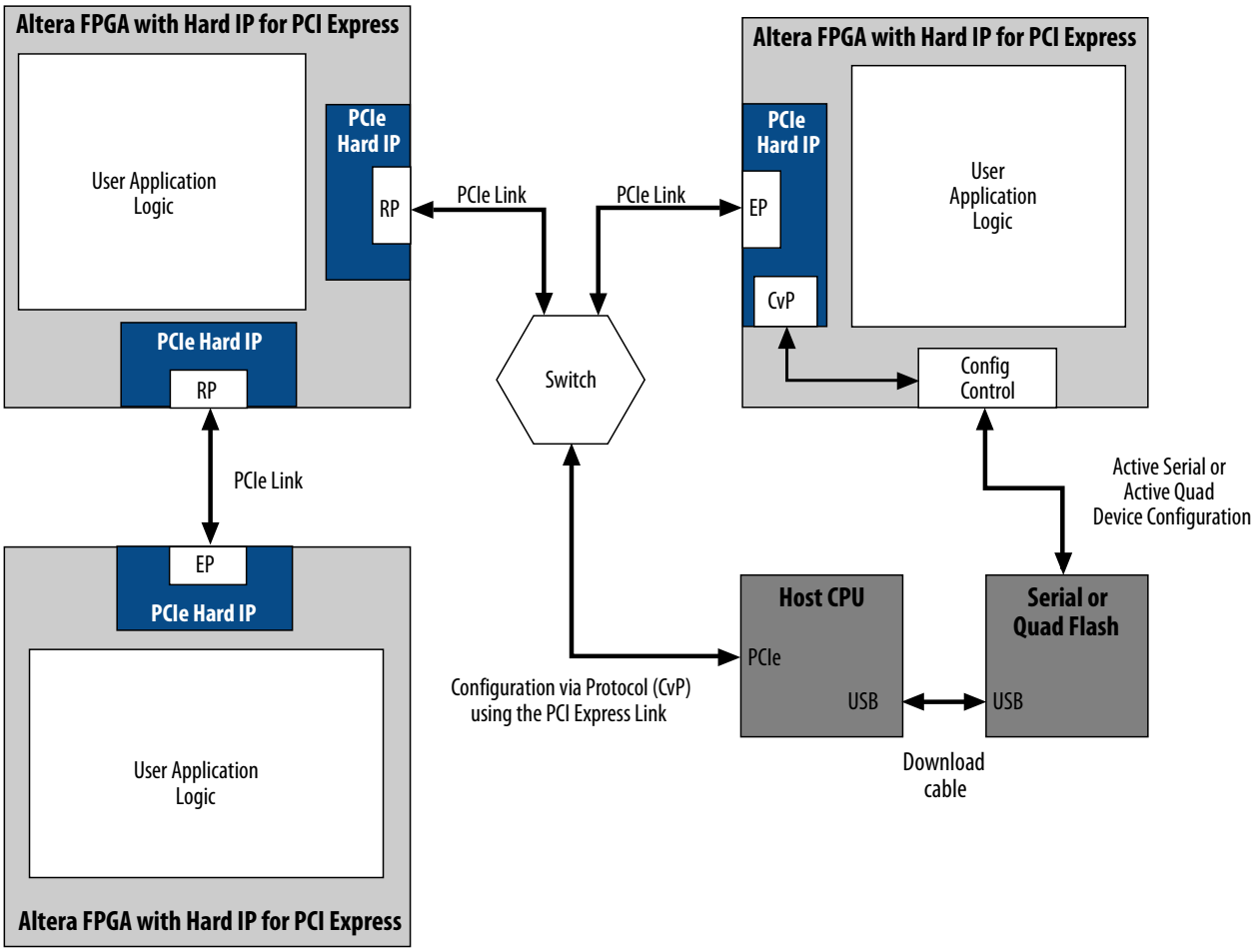


Figure 1-3: PCI Express Application Using Configuration via Protocol

The Arria V design below includes the following components:

- A Root Port that connects directly to a second FPGA that includes an Endpoint.
- Two Endpoints that connect to a PCIe switch.
- A host CPU that implements CvP using the PCI Express link connects through the switch. For more information about configuration over a PCI Express link below.



Related Information

Configuration via Protocol (CvP)Implementation in Altera FPGAs User Guide

Example Designs

The following example designs are available for the Avalon-MM Arria V Hard IP for PCI Express IP Core. You can download them from the ***<install_dir>*/ip/altera/altera_pcie/altera_pcie_*<dev>*__hip_avmm/example_designs** directory:

- **ep_g1x1.qsys**
- **ep_g1x4.qsys**
- **ep_g1x8.qsys**
- **ep_g2x1.qsys**
- **ep_g2x4.qsys**

Click on the link below to get started with the example design provided in this user guide.

Related Information

Getting Started with the Avalon-MM Arria V Hard IP for PCI Express on page 2-1

Debug Features

Debug features allow observation and control of the Hard IP for faster debugging of system-level problems.

Related Information

Debugging on page 13-1

IP Core Verification

To ensure compliance with the PCI Express specification, Altera performs extensive verification. The simulation environment uses multiple testbenches that consist of industry-standard bus functional models (BFMs) driving the PCI Express link interface. Altera performs the following tests in the simulation environment:

- Directed and pseudorandom stimuli are applied to test the Application Layer interface, Configuration Space, and all types and sizes of TLPs
- Error injection tests that inject errors in the link, TLPs, and Data Link Layer Packets (DLLPs), and check for the proper responses
- PCI-SIG® Compliance Checklist tests that specifically test the items in the checklist
- Random tests that test a wide range of traffic patterns

Altera provides the following two example designs that you can leverage to test your PCBs and complete compliance base board testing (CBB testing) at PCI-SIG.

Related Information

- **PCI SIG Gen3 x8 Merged Design - Stratix V**
- **PCI SIG Gen2 x8 Merged Design - Stratix V**

Compatibility Testing Environment

Altera has performed significant hardware testing to ensure a reliable solution. In addition, Altera internally tests every release with motherboards and PCI Express switches from a variety of manufacturers. All PCI-SIG compliance tests are run with each IP core release.

Performance and Resource Utilization

Because the PCIe protocol stack is implemented in hardened logic, it uses less than 1% of device resources.

The Avalon-MM bridge is implemented in soft logic and functions as a front end to the hardened protocol stack. The following table shows the typical device resource utilization for selected configurations using the current version of the Quartus II software. With the exception of M10K memory blocks, the numbers of ALMs and logic registers in the following tables are rounded up to the nearest 50.

Table 1-5: Performance and Resource Utilization Avalon-MM Hard IP for PCI Express

Data Rate or Interface Width	ALMs	Memory M10K	Logic Registers
Avalon-MM Bridge			
Gen1 ×4	1250	27	1700
Gen2 ×8	2100	35	3050
Avalon-MM Interface–Completer Only			
64	600	11	900
128	1350	22	2300
Avalon-MM–Completer Only Single DWord			
64	160	0	230

Note: Soft calibration of the transceiver module requires additional logic. The amount of logic required depends on the configuration.

Related Information

Fitter Resources Reports

Recommended Speed Grades

Table 1-6: Arria V Recommended Speed Grades for Link Widths and Application Layer Clock Frequencies

Altera recommends setting the Quartus II Analysis & Synthesis Settings **Optimization Technique** to **Speed** when the Application Layer clock frequency is 250 MHz. For information about optimizing synthesis, refer to *Setting Up and Running Analysis and Synthesis* in Quartus II Help. For more information about how to effect the **Optimization Technique** settings, refer to *Area and Timing Optimization* in volume 2 of the *Quartus II Handbook*. .

Link Rate	Link Width	Interface Width	Application Clock Frequency (MHz)	Recommended Speed Grades
Gen1	×1	64 bits	62.5[(2)],125	–4,–5,–6
	×2	64 bits	125	–4,–5,–6
	×4	64 bits	125	–4,–5,–6
	×8	128 bits	125	–4,–5,–6

(2) This is a power-saving mode of operation

Link Rate	Link Width	Interface Width	Application Clock Frequency (MHz)	Recommended Speed Grades
Gen2	×1	64 bits	125	–4,–5
	×2	64 bits	125	–4,–5
	×4	128 bits	125	–4,–5

Related Information

- **Area and Timing Optimization**
- **Altera Software Installation and Licensing Manual**
- **Setting up and Running Analysis and Synthesis**

Steps in Creating a Design for PCI Express

Before you begin

Select the PCIe variant that best meets your design requirements.

- Is your design an Endpoint or Root Port?
- What Generation do you intend to implement?
- What link width do you intend to implement?
- What bandwidth does your application require?
- Does your design require CvP?

1. Select parameters for that variant.
2. Simulate using an Altera-provided example design. All of Altera's PCI Express example designs are available under **<*install_dir*>/ip/altera/altera_pcie/**. Alternatively, create a simulation model and use your own custom or third-party BFM. The Qsys Generate menu generates simulation models. Altera supports ModelSim-Altera for all IP. The PCIe cores support the Aldec RivieraPro, Cadence NCsim, Mentor Graphics ModelSim, and Synopsys VCS and VCS-MX simulators.
3. Compile your design using the Quartus II software. If the versions of your design and the Quartus II software you are running do not match, regenerate your PCIe design.
4. Download your design to an Altera development board or your own PCB. Click on the *All Development Kits* link below for a list of Altera's development boards.
5. Test the hardware. You can use Altera's SignalTap® II Logic Analyzer or a third-party protocol analyzer to observe behavior.
6. Substitute your Application Layer logic for the Application Layer logic in Altera's testbench. Then repeat Steps 3–6. In Altera's testbenches, the PCIe core is typically called the DUT (device under test). The Application Layer logic is typically called APPS.

Related Information

- **Parameter Settings** on page 3-1
- **Getting Started with the Avalon-MM Arria V Hard IP for PCI Express**
- **All Development Kits**

Getting Started with the Avalon-MM Arria V Hard IP for PCI Express

2

2014.12.15

UG-01105_avmm

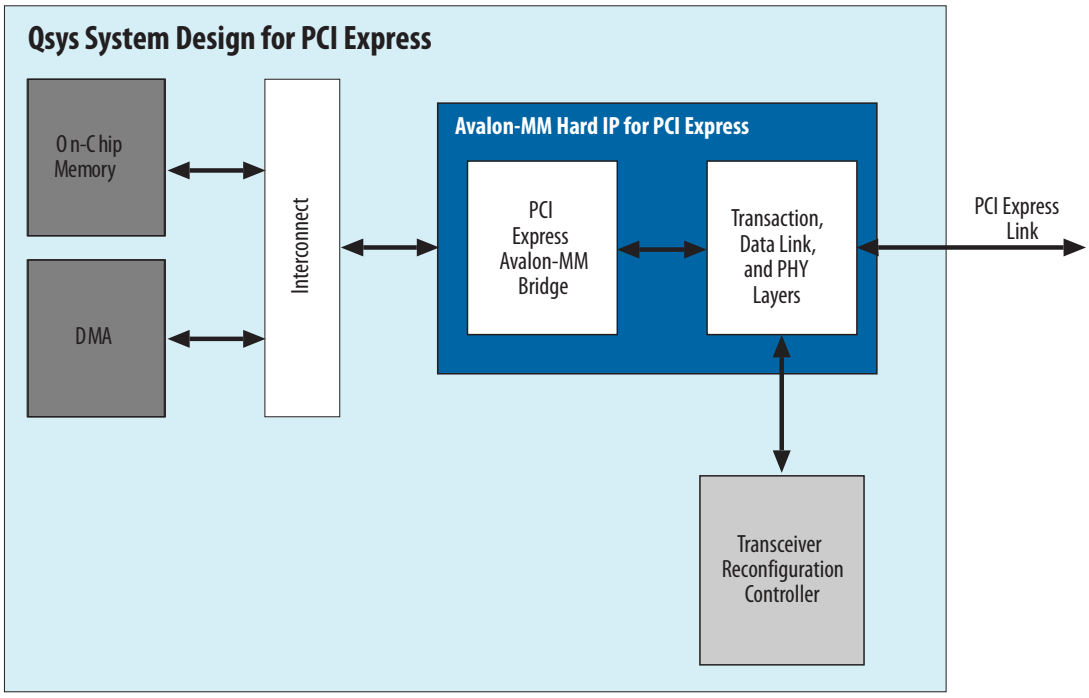
Subscribe Send Feedback

You can download a design example for the Avalon-MM Arria V Hard IP for PCI Express from the **<*install_dir*>/ip/altera/altera_pcie/altera_pcie-<*dev*>_hip_avmm/example_designs** directory. This walkthrough uses the a Gen1 x4 Endpoint, **ep_g1x4.qsys**.

The design examples contain the following components:

- Avalon-MM Arria V Hard IP for PCI Express IP core
- On-Chip memory
- DMA controller
- Transceiver Reconfiguration Controller
- Two Avalon-MM pipeline bridges

Figure 2-1: Qsys Generated Endpoint



© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

The design example transfers data between an on-chip memory buffer located on the Avalon-MM side and a PCI Express memory buffer located on the root complex side. The data transfer uses the DMA component which is programmed by the PCI Express software application running on the Root Complex processor.

The example design also includes the Transceiver Reconfiguration Controller which allows you to dynamically reconfigure transceiver settings. This component is necessary for high performance transceiver designs.

Related Information

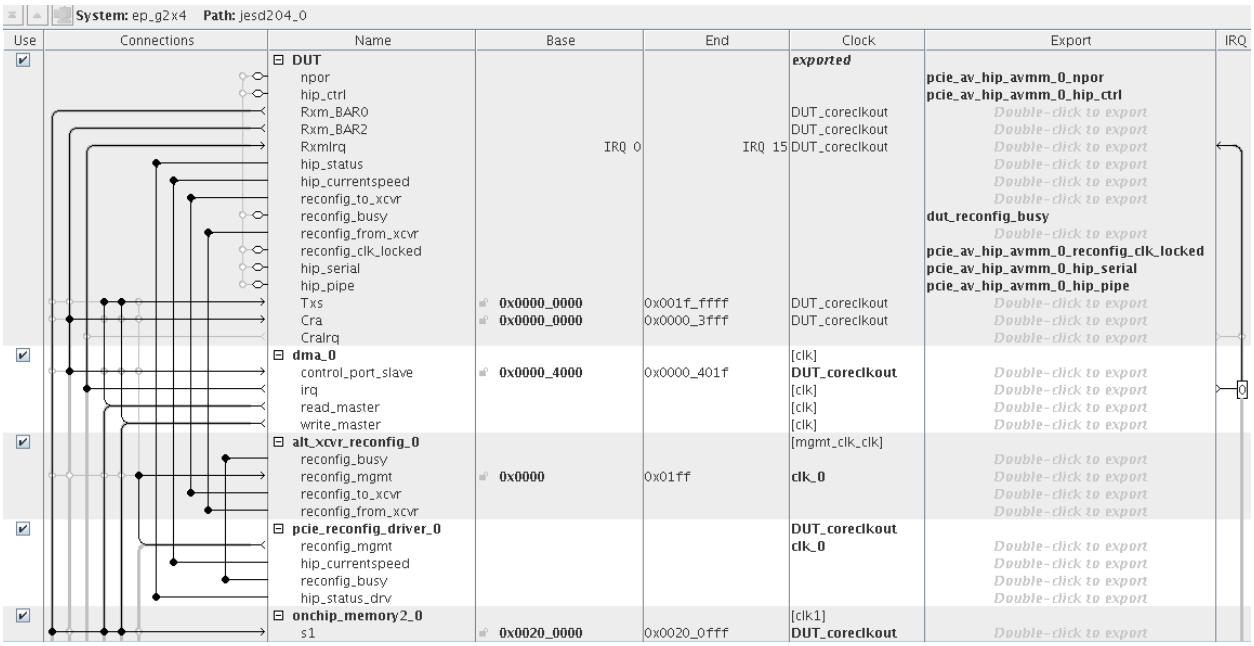
- **Generating the Example Design** on page 2-3
- **Creating a System with Qsys**
  This document provides an introduction to Qsys.

Running Qsys

1. Choose **Programs > Altera > Quartus II><version_number>** (Windows Start menu) to run the Quartus II software. Alternatively, you can also use the Quartus II Web Edition software.
2. On the File menu, select **New**, then **Qsys System File**.
3. Open the **ep_g1x4.qsys** example design.

The following figure shows a Qsys system that includes the Transceiver Reconfiguration Controller and the Altera PCIe Reconfig Driver IP Cores. The Transceiver Reconfiguration Controller performs dynamic reconfiguration of the analog transceiver settings to optimize signal quality. You must include these components to the Qsys system to run successfully in hardware.

Figure 2-2: Qsys Avalon-MM Design for PCIe with Transceiver Reconfiguration Components



Refer to *Creating a System with Qsys* in volume 1 of the *Quartus II Handbook* for more information about how to use Qsys. For an explanation of each Qsys menu item, refer to *About Qsys* in Quartus II Help.

Related Information

- **Creating a System with Qsys**
- **About Qsys**

Generating the Example Design

1. On the Generate menu, select **Generate Testbench System**. The **Generation** dialog box appears.
2. Under **Testbench System**, set the following options:
   a. For **Create testbench Qsys system**, select **Standard, BFMs for standard Qsys interfaces**.
   b. For **Create testbench simulation model**, select **Verilog**.
3. You can retain the default values for all other parameters.
4. Click **Generate**.
5. After Qsys reports **Generation Completed**, click **Close**.
6. On the File menu, click **Save**.

The following table lists the testbench and simulation directories Qsys generates.

Table 2-1: Qsys System Generated Directories

Directory	Location
Qsys system	**<*project_dir*>/ep_g1x4**
Testbench	**<*project_dir*>/ep_g1x4/testbench/<*cad_vendor*>**
Simulation Model	**<*project_dir*>/ep_g1x4/testbench/ep_g2x4_tb/simulation/**

The design example simulation includes the following components and software:

- The Qsys system
- A testbench. You can view this testbench in Qsys by opening **<*project_dir*>/ep_g2x4/testbench/ep_g1x4_tb.qsys**.
- The ModelSim software

Note: You can also use any other supported third-party simulator to simulate your design.

Complete the following steps to run the Qsys testbench:

1. In a terminal window, change to the **<*project_dir*>/ep_g1x4/testbench/mentor** directory.
2. Start the ModelSim® simulator.
3. Type the following commands in a terminal window:
   a. `do msim_setup.tcl`
   b. `ld_debug`
   c. `run 140000 ns`

The driver performs the following transactions with status of the transactions displayed in the ModelSim simulation message window:

1. Various configuration accesses to the Avalon-MM Arria V Hard IP for PCI Express in your system after the link is initialized
2. Setup of the Address Translation Table for requests that are coming from the DMA component
3. Setup of the DMA controller to read 512 Bytes of data from the Transaction Layer Direct BFM shared memory
4. Setup of the DMA controller to write the same data back to the Transaction Layer Direct BFM shared memory
5. Data comparison and report of any mismatch

Related Information

Simulating Altera Designs

Running A Gate-Level Simulation

The PCI Express testbenches run simulations at the register transfer level (RTL). However, it is possible to create you own gate-level simulations. Contact your Altera Sales Representative for instructions and an example that illustrate how to create a gate-level simulation from the RTL testbench.

Simulating the Single DWord Design

You can use the same testbench to simulate the Completer-Only Single Dword IP core by changing the settings in the driver file.

1. In a terminal window, change to the ***<project_dir>*/<*variant*>/testbench/<*variant*>_tb/simulation/submodules** directory.
2. Open **altpcietb_bfm_driver_avmm.v** in your text editor.
3. To enable target memory tests and specify the completer-only single dword variant, specify the following parameters:
   a. `parameter RUN_TGT_MEM_TST = 1;`
   b. `parameter RUN_DMA_MEM_TST = 0;`
   c. `parameter AVALON_MM_LITE = 1;`
4. Change to the ***<project_dir>/variant*/testbench/mentor** directory.
5. Start the ModelSim simulator.
6. To run the simulation, type the following commands in a terminal window:
   a. `do msim_setup.tcl`
   b. `ld_debug` (The debug suffix stops optimizations, improving visibility in the ModelSim waveforms.)
   c. `run 140000 ns`

Understanding Channel Placement Guidelines

Arria V transceivers are organized in banks. The transceiver bank boundaries are important for clocking resources, bonding channels, and fitting. Refer to the channel placement figures following *Serial Interface Signals* for illustrations of channel placement.

Generating Quartus II Synthesis Files

1. On the **Generate** menu, select **Generate HDL**.
2. For **Create HDL design files for synthesis**, select **Verilog**.
   You can leave the default settings for all other items.
3. Click **Generate** to generate files for Quartus II synthesis.
4. Click **Finish** when the generation completes.

Compiling the Design in the Quartus II Software

To compile the Qsys design example in the Quartus II software, you must create a Quartus II project and add your Qsys files to that project.

Complete the following steps to create your Quartus II project:

1. Click the **New Project Wizard** icon.
2. Click **Next** in the **New Project Wizard: Introduction** (The introduction does not appear if you previously turned it off)
3. On the **Directory, Name, Top-Level Entity** page, enter the following information:
   a. The working directory shown is correct. You do not have to change it.
   b. For the project name, browse to the synthesis directory that includes your Qsys project, **<*working_dir*>/ep_g1x4/synthesis**. Select your variant name, **ep_g1x4.v**. Then, click **Open.**
   c. If the top-level design entity and Qsys system names are identical, the Quartus II software treats the Qsys system as the top-level design entity.
4. Click **Next** to display the **Add Files** page.
5. Complete the following steps to add the Quartus II IP File (**.qip**)to the project:
   a. Click the **browse** button. The **Select File** dialog box appears.
   b. In the **Files of type** list, select **IP Variation Files (*.qip)**.
   c. Browse to the **<*working_dir*>/ep_g1x4/synthesis** directory.
   d. Click **ep_g1x4.qip** and then click **Open**.
   e. On the **Add Files** page, click **Add**, then click **OK**.
6. Click **Next** to display the **Device** page.
7. On the **Family & Device Settings** page, choose the following target device family and options:
   a. In the **Family** list, select Arria V (GT/GX/ST/SX).
   b. In the **Devices** list, select Arria V **GX Extended Features.**.
   c. In the **Available Devices** list, select **5AGXFB3H6F35C6**.
8. Click **Next** to close this page and display the **EDA Tool Settings** page.

9. From the **Simulation** list, select **ModelSim®**. From the **Format** list, select the HDL language you intend to use for simulation.
10. Click **Next** to display the **Summary** page.
11. Check the **Summary** page to ensure that you have entered all the information correctly.
12. Click **Finish** to create the Quartus II project.
13. Add the Synopsys Design Constraint (SDC) commands shown in the following example to the top-level design file for your Quartus II project.
14. To compile your design using the Quartus II software, on the Processing menu, click **Start Compilation**. The Quartus II software then performs all the steps necessary to compile your design.
15. After compilation, expand the **TimeQuest Timing Analyzer** folder in the Compilation Report. Note whether the timing constraints are achieved in the Compilation Report.
16. If your design does not initially meet the timing constraints, you can find the optimal Fitter settings for your design by using the Design Space Explorer. To use the Design Space Explorer, click **Launch Design Space Explorer** on the tools menu.

Example 2-1: Synopsys Design Constraints

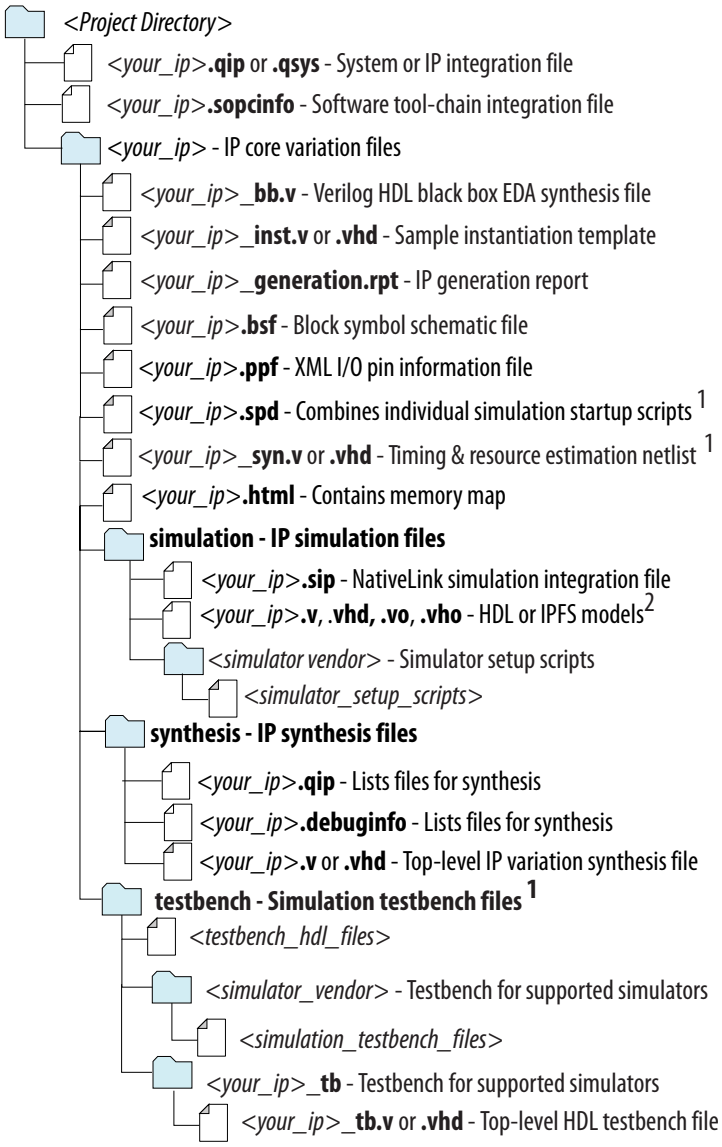
```
create_clock -period “100 MHz” -name {refclk_pci_express}{*refclk_*}
derive_pll_clocks
derive_clock_uncertainty

# PHY IP reconfig controller constraints
# Set reconfig_xcvr clock
# Modify to match the actual clock pin name
# used for this clock, and also changed to have the correct period set
create_clock -period "125 MHz" -name {reconfig_xcvr_clk}{*reconfig_xcvr_clk*}
```

Files Generated for Altera IP Cores

Figure 2-3: IP Core Generated Files

The Quartus II software generates the following output for your IP core.



Notes:
1. If supported and enabled for your IP variation
2. If functional simulation models are generated

Programming a Device

After you compile your design, you can program your targeted Altera device and verify your design in hardware.

For more information about programming Altera FPGAs, refer to *Quartus II Programmer*.

Related Information

Quartus II Programmer

Parameter Settings 3

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

Avalon-MM System Settings

Table 3-1: System Settings for PCI Express

Parameter	Value	Description
Number of Lanes	**×1, ×2, ×4, ×8**	Specifies the maximum number of lanes supported.
Lane Rate	**Gen1 (2.5 Gbps)** **Gen2 (2.5/5.0 Gbps)**	Specifies the maximum data rate at which the link can operate.
Port type	**Root Port** **Native Endpoint**	Specifies the port type. Altera recommends **Native Endpoint** for all new Endpoint designs. The **Legacy Endpoint** is not available for the Avalon-MM Arria V Hard IP for PCI Express. The Endpoint stores parameters in the Type 0 Configuration Space. The Root Port stores parameters in the Type 1 Configuration Space.
RX Buffer credit allocation - performance for received requests	**Minimum** **Low** **Balanced**	Determines the allocation of posted header credits, posted data credits, non-posted header credits, completion header credits, and completion data credits in the 16 KByte RX buffer. The 5 settings allow you to adjust the credit allocation to optimize your system. The credit allocation for the selected setting displays in the message pane. Refer to the *Throughput Optimization* chapter for more information about optimizing performance. The Flow Control chapter explains how the **RX credit allocation** and the **Maximum payload RX Buffer credit allocation** and the **Maximum payload size** that you choose affect the allocation of flow control credits. You can set the **Maximum payload size** parameter on the **Device** tab.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Parameter	Value	Description
		The **Message** window of the GUI dynamically updates the number of credits for Posted, Non-Posted Headers and Data, and Completion Headers and Data as you change this selection. • **Minimum RX Buffer credit allocation -performance for received requests)**–This setting configures the minimum PCIe specification allowed for non-posted and posted request credits, leaving most of the RX Buffer space for received completion header and data. Select this option for variations where application logic generates many read requests and only infrequently receives single requests from the PCIe link. • **Low**–This setting configures a slightly larger amount of RX Buffer space for non-posted and posted request credits, but still dedicates most of the space for received completion header and data. Select this option for variations where application logic generates many read requests and infrequently receives small bursts of requests from the PCIe link. This option is recommended for typical endpoint applications where most of the PCIe traffic is generated by a DMA engine that is located in the endpoint application layer logic. • **Balanced**–This setting allocates approximately half the RX Buffer space to received requests and the other half of the RX Buffer space to received completions. Select this option for variations where the received requests and received completions are roughly equal.
Reference clock frequency	**100 MHz** **125 MHz**	The *PCI Express Base Specification* requires a 100 MHz ±300 ppm reference clock. The 125 MHz reference clock is provided as a convenience for systems that include a 125 MHz clock source.
Use 62.5 MHz application clock	**On/Off**	This mode is only available only for Gen1 ×1.
Enable configuration via PCIe link	**On/Off**	When **On**, the Quartus II software places the Endpoint in the location required for configuration via protocol (CvP). For more information about CvP, click the *Configuration via Protocol (CvP)* link below. CvP is not supported for Gen3 variants.

Related Information

PCI Express Base Specification 2.1 or 3.0

Base Address Register (BAR) Settings

You can configure up to six 32-bit BARs or three 64-bit BARs.

Table 3-2: BAR Registers

Parameter	Value	Description
Type	**Disabled** **64-bit prefetchable memory** **32-bit non-prefetchable memory** **32-bit prefetchable memory** **I/O address space**	Defining memory as prefetchable allows data in the region to be fetched ahead anticipating that the requestor may require more data from the same region than was originally requested. If you specify that a memory is prefetchable, it must have the following 2 attributes: • Reads do not have side effects • Write merging is allowed The **32-bit prefetchable memory** and **I/O address space** BARs are only available for the **Legacy Endpoint**.
Size	Not configurable	Specifies the memory size calculated from other parameters you enter.

Table 3-3: Device ID Registers

The following table lists the default values of the read-only Device ID registers. You can use the parameter editor to change the values of these registers. Refer to *Type 0 Configuration Space Registers* for the layout of the Device Identification registers.

Register Name	Range	Default Value	Description
Vendor ID	16 bits	0x00000000	Sets the read-only value of the `Vendor ID` register. This parameter cannot be set to 0xFFFF, per the *PCI Express Specification*. Address offset: 0x000.
Device ID	16 bits	0x00000001	Sets the read-only value of the `Device ID` register. This register is only valid in the Type 0 (Endpoint) Configuration Space. Address offset: 0x000.

Register Name	Range	Default Value	Description
Revision ID	8 bits	0x00000001	Sets the read-only value of the `Revision ID` register. Address offset: 0x008.
Class code	24 bits	0x00000000	Sets the read-only value of the `Class Code` register. Address offset: 0x008.
Subsystem Vendor ID	16 bits	0x00000000	Sets the read-only value of the `Subsystem Vendor ID` register in the PCI Type 0 Configuration Space. This parameter cannot be set to 0xFFFF per the *PCI Express Base Specification*. This value is assigned by PCI-SIG to the device manufacturer. This register is only valid in the Type 0 (Endpoint) Configuration Space. Address offset: 0x02C.
Subsystem Device ID	16 bits	0x00000000	Sets the read-only value of the `Subsystem Device ID` register in the PCI Type 0 Configuration Space. Address offset: 0x02C

Related Information

PCI Express Base Specification 2.1 or 3.0

Device Capabilities

Table 3-4: Capabilities Registers

Parameter	Possible Values	Default Value	Description
Maximum payload size	**128 bytes** **256 bytes**	128 bytes	Specifies the maximum payload size supported. This parameter sets the read-only value of the max payload size supported field of the Device Capabilities register (0x084[2:0]). Address: 0x084**.**
Completion timeout range	**ABCD** **BCD** **ABC** **AB** **B** **A** **None**	ABCD	Indicates device function support for the optional completion timeout programmability mechanism. This mechanism allows system software to modify the completion timeout value. This field is applicable only to Root Ports and Endpoints that issue requests on their own behalf. Completion timeouts are specified and enabled in the Device Control 2 register (0x0A8) of the *PCI Express Capability Structure Version*. For all other

Parameter	Possible Values	Default Value	Description
			functions this field is reserved and must be hardwired to 0x0000b. Four time value ranges are defined: • Range A: 50 us to 10 ms • Range B: 10 ms to 250 ms • Range C: 250 ms to 4 s • Range D: 4 s to 64 s Bits are set to show timeout value ranges supported. The function must implement a timeout value in the range 50 s to 50 ms. The following values specify the range: • None – Completion timeout programming is not supported • 0001 Range A • 0010 Range B • 0011 Ranges A and B • 0110 Ranges B and C • 0111 Ranges A, B, and C • 1110 Ranges B, C and D • 1111 Ranges A, B, C, and D All other values are reserved. Altera recommends that the completion timeout mechanism expire in no less than 10 ms.
Implement completion timeout disable	**On/Off**	On	For Endpoints using PCI Express version 2.1 or 3.0, this option must be **On**. The timeout range is selectable. When **On**, the core supports the completion timeout disable mechanism via the PCI Express `Device Control Register 2`. The Application Layer logic must implement the actual completion timeout mechanism for the required ranges.

Error Reporting

Table 3-5: Error Reporting

Parameter	Value	Default Value	Description
Advanced error reporting (AER)	**On/Off**	Off	When **On**, enables the Advanced Error Reporting (AER) capability.

Parameter	Value	Default Value	Description
ECRC checking	**On/Off**	Off	When **On**, enables ECRC checking. Sets the read-only value of the ECRC check capable bit in the `Advanced Error Capabilities and Control Register`. This parameter requires you to enable the AER capability.
ECRC generation	**On/Off**	Off	When **On**, enables ECRC generation capability. Sets the read-only value of the ECRC generation capable bit in the `Advanced Error Capabilities and Control Register`. This parameter requires you to enable the AER capability. Not applicable for Avalon-MM DMA.

Link Capabilities

Table 3-6: Link Capabilities

Parameter	Value	Description
Link port number	**0x01**	Sets the read-only value of the port number field in the `Link Capabilities Register`.
Slot clock configuration	**On/Off**	When **On**, indicates that the Endpoint or Root Port uses the same physical reference clock that the system provides on the connector. When **Off**, the IP core uses an independent clock regardless of the presence of a reference clock on the connector.

MSI and MSI-X Capabilities

Table 3-7: MSI and MSI-X Capabilities

Parameter	Value	Description
MSI messages requested	**1, 2, 4, 8, 16**	Specifies the number of messages the Application Layer can request. Sets the value of the `Multiple Message Capable` field of the `Message Control` register, 0x050[31:16].
MSI-X Capabilities		
Implement MSI-X	**On/Off**	When **On**, enables the MSI-X functionality.
	Bit Range	

Parameter	Value	Description
Table size	[10:0]	System software reads this field to determine the MSI-X Table size <*n*>, which is encoded as <*n–1*>. For example, a returned value of 2047 indicates a table size of 2048. This field is read-only. Legal range is 0–2047 (2^{11}). Address offset: 0x068[26:16]
Table Offset	[31:0]	Points to the base of the MSI-X Table. The lower 3 bits of the table BAR indicator (BIR) are set to zero by software to form a 32-bit qword-aligned offset. This field is read-only.
Table BAR Indicator	[2:0]	Specifies which one of a function's BARs, located beginning at 0x10 in Configuration Space, is used to map the MSI-X table into memory space. This field is read-only. Legal range is 0–5.
Pending Bit Array (PBA) Offset	[31:0]	Used as an offset from the address contained in one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower 3 bits of the PBA BIR are set to zero by software to form a 32-bit qword-aligned offset. This field is read-only.
PBA BAR Indicator	[2:0]	Specifies the function Base Address registers, located beginning at 0x10 in Configuration Space, that maps the MSI-X PBA into memory space. This field is read-only. Legal range is 0–5.

Power Management

Table 3-8: Power Management Parameters

Parameter	Value	Description
Endpoint L0s acceptable latency	**Maximum of 64 ns** **Maximum of 128 ns** **Maximum of 256 ns** **Maximum of 512 ns** **Maximum of 1 us** **Maximum of 2 us** **Maximum of 4 us** **No limit**	This design parameter specifies the maximum acceptable latency that the device can tolerate to exit the L0s state for any links between the device and the root complex. It sets the read-only value of the Endpoint L0s acceptable latency field of the `Device Capabilities Register` (0x084). This Endpoint does not support the L0s or L1 states. However, in a switched system there may be links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports. The default value of this parameter is 64 ns. This is the safest setting for most designs.
Endpoint L1 acceptable latency	**Maximum of 1 us** **Maximum of 2 us** **Maximum of 4 us** **Maximum of 8 us** **Maximum of 16 us** **Maximum of 32 us** **No limit**	This value indicates the acceptable latency that an Endpoint can withstand in the transition from the L1 to L0 state. It is an indirect measure of the Endpoint's internal buffering. It sets the read-only value of the Endpoint L1 acceptable latency field of the `Device Capabilities Register`. This Endpoint does not support the L0s or L1 states. However, a switched system may include links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports. The default value of this parameter is 1 µs. This is the safest setting for most designs.

Avalon Memory-Mapped System Settings

Table 3-9: Avalon Memory-Mapped System Settings

Parameter	Value	Description
Avalon-MM data width	**64-bit** **128-bit**	Specifies the data width for the Application Layer to Transaction Layer interface. Refer to *Application Layer Clock Frequencies for All Combinations of Link Width, Data Rate and Application Layer Interface Widths* for all legal combinations of data width, number of lanes, Application Layer clock frequency, and data rate.
Avalon-MM address width	**32-bit** **64-bit**	Specifies the address width for Avalon-MM RX master ports that access Avalon-MM slaves in the Avalon address domain. When you select 32-bit addresses, the PCI Express Avalon-MM Bridge performs address translation. When you specify 64-bits addresses, no address translation is performed in either direction. The destination address specified is forwarded to the Avalon-MM interface without any changes. For the Avalon-MM interface with DMA, this value must be set to **64**.
Peripheral mode	**Requester/ Completer** **Completer-Only**	Specifies whether the Avalon-MM Arria V Hard IP for PCI Express is capable of sending requests to the upstream PCI Express devices, and whether the incoming requests are pipelined. **Requester/Completer**—In this mode, the Hard IP can send request packets on the PCI Express TX link and receive request packets on the PCI Express RX link. **Completer-Only**—In this mode, the Hard IP can receive requests, but cannot initiate upstream requests. However, it can transmit completion packets on the PCI Express TX link. This mode removes the Avalon-MM TX slave port and thereby reduces logic utilization.

Parameter	Value	Description
Single DW Completer	**On/Off**	This is a non-pipelined version of **Completer Only** mode. At any time, only a single request can be outstanding. **Single dword completer** uses fewer resources than **Completer Only**. This variant is targeted for systems that require simple read and write register accesses from a host CPU. If you select this option, the width of the data for RXM BAR masters is always 32 bits, regardless of the **Avalon-MM** width. For the Avalon-MM interface with DMA, this value must be **Off** .
Control register access (CRA) Avalon-MM slave port	**On/Off**	Allows read and write access to bridge registers from the interconnect fabric using a specialized slave port. This option is required for **Requester/Completer** variants and optional for **Completer Only** variants. Enabling this option allows read and write access to bridge registers, except in the Completer-Only single dword variations.
Enable multiple MSI/MSI-X support	**On/Off**	When you turn this option **On**, the core exports top-level MSI and MSI-X interfaces that you can use to implement a Customer Interrupt Handler for MSI and MSI-X interrupts. For more information about the Custom Interrupt Handler, refer to *Interrupts for End Points Using the Avalon-MM Interface with Multiple MSI/MSI-X Support*. If you turn this option **Off**, the core handles interrupts internally.
Auto enabled PCIe interrupt (enabled at power-on)	**On/Off**	Turning on this option enables the Avalon-MM Arria V Hard IP for PCI Express interrupt register at power-up. Turning off this option disables the interrupt register at power-up. The setting does not affect run-time configuration of the interrupt enable register. For the Avalon-MM interface with DMA, this value must be **Off**.

Parameter	Value	Description
Enable hard IP status bus	**On/Off**	When you turn this option on, your top-level variant includes the signals necessary to connect to the Transceiver Reconfiguration Controller IP Core, your variant, including: • Link status signals • ECC error signals • TX and RX parity error signals • Completion header and data signals, indicating the total number of Completion TLPs currently stored in the RX buffer Altera recommends that you include the Transceiver Reconfiguration Controller IP Core in your design to improve signal quality.
Enable hard IP status extension bus	**On/Off**	When you turn this option on, your top-level variant includes signals that are useful for debugging, including link training and status, error, and the Transaction Layer Configuration Space signals. The top-level variant also includes signals showing the start and end of packets, error, ready, and BAR signals for the native Avalon-ST interface that connects to the Transaction Layer. The following signals are included in the top-level variant: • Link status signals • ECC error signals • Transaction Layer Configuration Space signals • Avalon-ST packet, error, ready, and BAR signals
Avalon to PCIe Address Translation Settings		
Number of address pages	**1, 2, 4, 8, 16, 32, 64, 128, 256, 512**	Specifies the number of pages required to translate Avalon-MM addresses to PCI Express addresses before a request packet is sent to the Transaction Layer. Each of the 512 possible entries corresponds to a base address of the PCI Express memory segment of a specific size. This parameter is only necessary when you select 32-bit addressing.
Size of address pages	**4 KBytes–4 GBytes**	Specifies the size of each memory segment. Each memory segment must be the same size. Refer to *Avalon-MM-to-PCI Express Address Translation Algorithm for 32-Bit Bridge* for more information about address translation. This parameter is only necessary when you select 32-bit addressing.

Interfaces and Signal Descriptions

4

2014.12.15

UG-01105_avmm

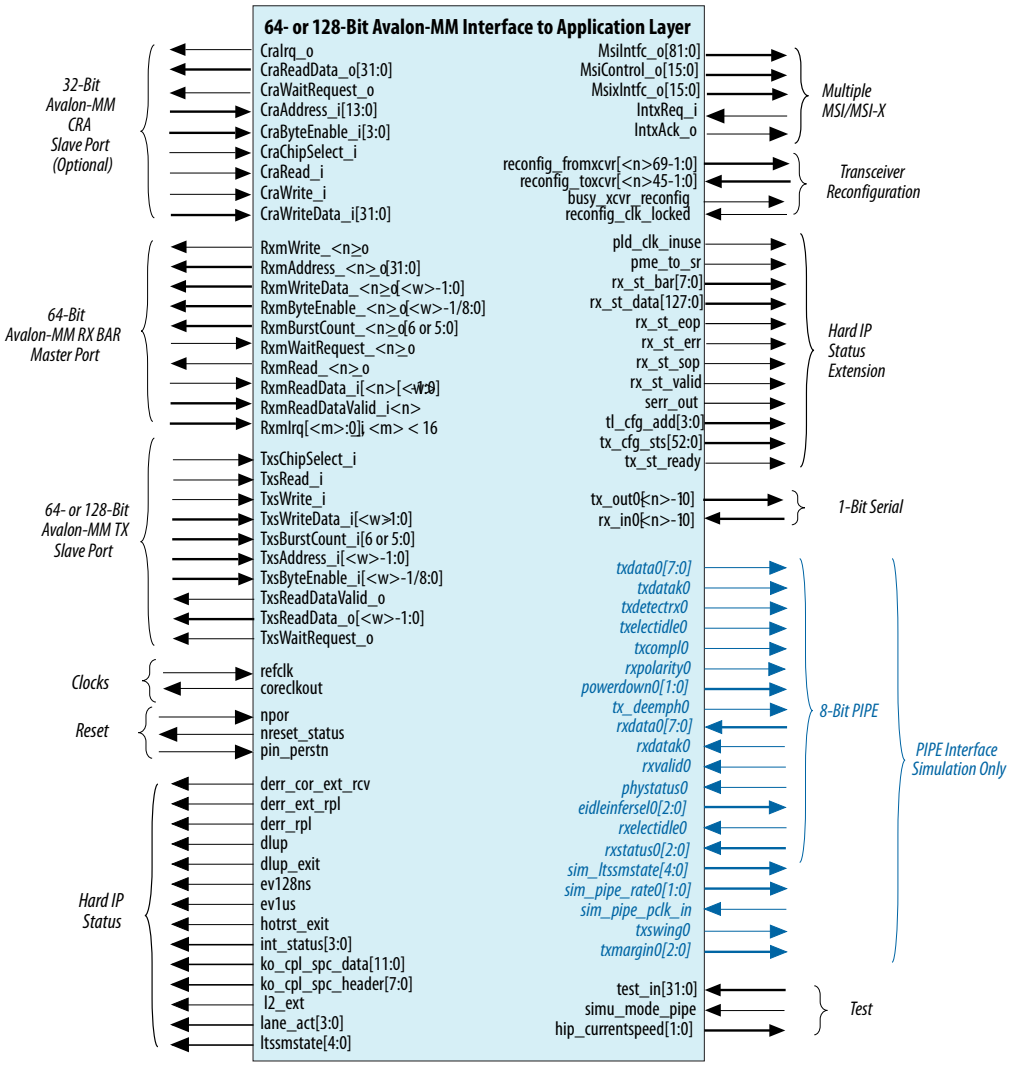
Subscribe Send Feedback

64- or 128-Bit Avalon-MM Interface to the Application Layer

This chapter describes the top-level signals of the Arria V Hard IP for PCI Express using the Avalon-MM interface to the Application Layer. The Avalon-MM bridge translates PCI Express read, write and completion TLPs into standard Avalon-MM read and write commands for the Avalon-MM RX Master Port interface. For the Avalon-MM TX Slave Port interface, the bridge translates Avalon-MM reads and writes into PCI Express TLPs. The Avalon-MM read and write commands are the same as those used by master and slave interfaces to access memories and registers. Consequently, you do not need a detailed understanding of the PCI Express TLPs to use this Avalon-MM variant.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered



Note: Signals listed for BAR0 are the same as those for BAR1–BAR5 when those BARs are enabled in the parameter editor.

Variations using the Avalon-MM interface implement the Avalon-MM protocol described in the *Avalon Interface Specifications*. Refer to this specification for information about the Avalon-MM protocol, including timing diagrams.

Related Information

Avalon Interface Specifications

32-Bit Non-Bursting Avalon-MM Control Register Access (CRA) Slave Signals

The optional CRA port for the full-featured IP core allows upstream PCI Express devices and external Avalon-MM masters to access internal control and status registers.

Table 4-1: Avalon-MM CRA Slave Interface Signals

Signal Name	Direction	Description
CraIrq_o	Output	Interrupt request. A port request for an Avalon-MM interrupt.
CraReadData_o[31:0]	Output	Read data lines
CraWaitRequest_o	Output	Wait request to hold off more requests
CraAddress_i[13:0]	Input	An address space of 16,384 bytes is allocated for the control registers. Avalon-MM slave addresses provide address resolution down to the width of the slave data bus. Because all addresses are byte addresses, this address logically goes down to bit 2. Bits 1 and 0 are 0.
CraByteEnable_i[3:0]	Input	Byte enable
CraChipSelect_i	Input	Chip select signal to this slave
CraRead_i	Input	Read enable
CraWrite_i	Input	Write request
CraWriteData_i[31:0]	Input	Write data

RX Avalon-MM Master Signals

This Avalon-MM master port propagates PCI Express requests to the Qsys interconnect fabric. For the full-feature IP core it propagates requests as bursting reads or writes. A separate Avalon-MM master port corresponds to each BAR.

Table 4-2: Avalon-MM RX Master Interface Signals

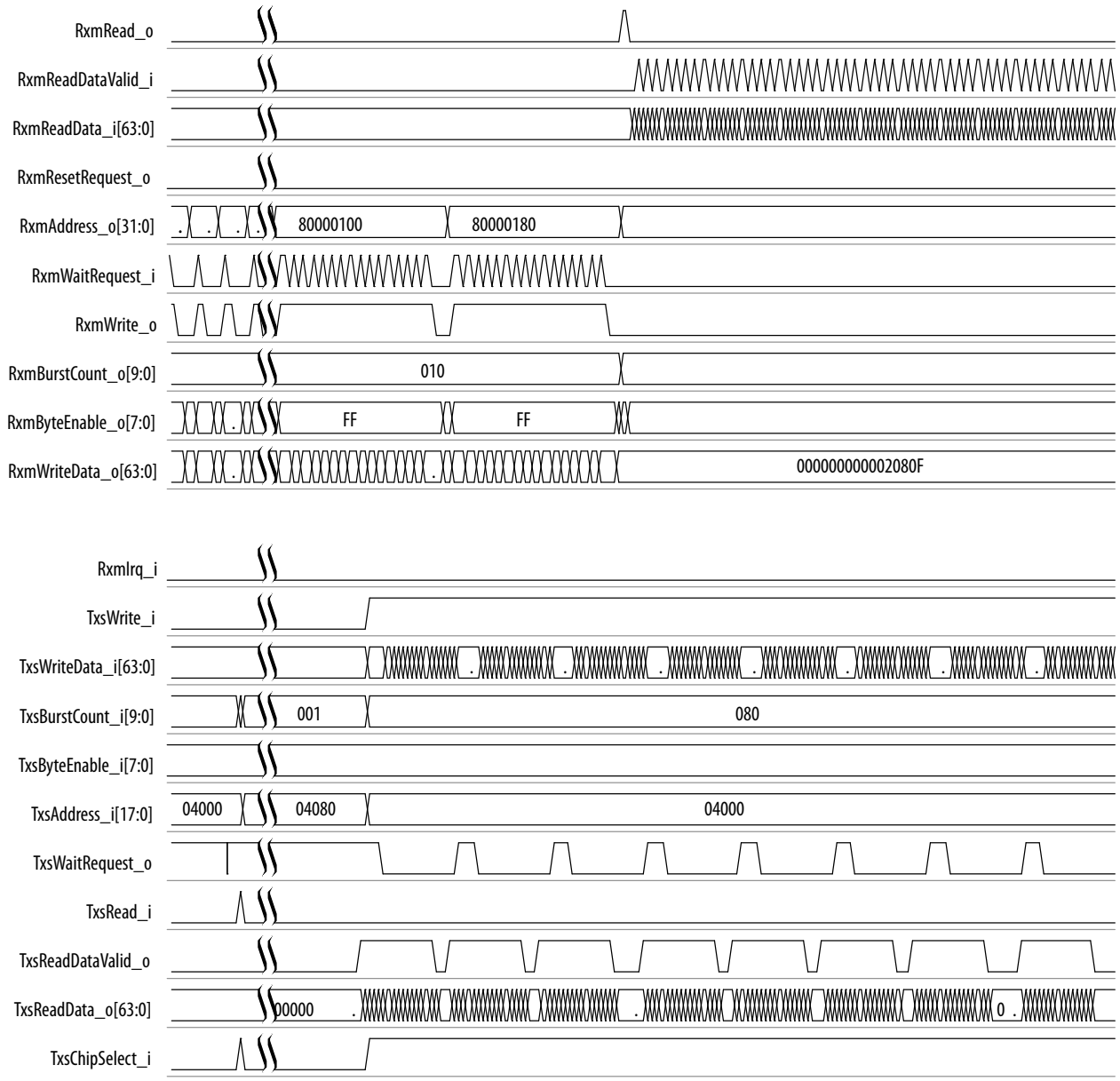
Signals that include Bar number 0 also exist for BAR1–BAR5 when additional BARs are enabled.

Signal Name	Direction	Description
RxmWrite<n>	Output	Asserted by the core to request a write to an Avalon-MM slave.
RxmAddress_<*n*>_o[31:0]	Output	The address of the Avalon-MM slave being accessed.
RxmWriteData_<*n*>_o[<*w*>-1:0]	Output	RX data being written to slave. <*w*> = 64 or 128 for the full-featured IP core. <*w*> = 32 for the completer-only IP core.

Signal Name	Direction	Description
RxmByteEnable_<*n*>_o[<*w*>-1:0]	Output	Byte enable for write data.
RXMBurstCount_<*n*>_o[6 or 5:0]	Output	The burst count, measured in qwords, of the RX write or read request. The width indicates the maximum data that can be requested. The maximum data in a burst is 512 bytes.
RXMWaitRequest_<*n*>_o	Input	Asserted by the external Avalon-MM slave to hold data transfer.
RXMRead_<*n*>_o	Output	Asserted by the core to request a read.
RXMReadData_<*n*>_o[<*w*>-1:0]	Input	Read data returned from Avalon-MM slave in response to a read request. This data is sent to the IP core through the TX interface. <*w*> = 64 or 128 for the full-featured IP core. <*w*> = 32 for the completer-only IP core.
RXMReadDataValid_<*n*>_i	Input	Asserted by the system interconnect fabric to indicate that the read data on is valid.
RxmIrq_<n>[<*m*>:0], <*m*>< 16	Input	Indicates an interrupt request asserted from the system interconnect fabric. This signal is only available when the CRA port is enabled. Qsys-generated variations have as many as 16 individual interrupt signals (<*m*>≤15). If rxm_irq_<n>[<*m*>:0] is asserted on consecutive cycles without the deassertion of all interrupt inputs, no MSI message is sent for subsequent interrupts. To avoid losing interrupts, software must ensure that all interrupt sources are cleared for each MSI message received.

The following figure illustrates the RX master port propagating requests to the Application Layer and also shows simultaneous, DMA read and write activity

Figure 4-1: Simultaneous DMA Read, DMA Write, and Target Access



64- or 128-Bit Bursting TX Avalon-MM Slave Signals

This optional Avalon-MM bursting slave port propagates requests from the interconnect fabric to the full-featured Avalon-MM Arria V Hard IP for PCI Express. Requests from the interconnect fabric are translated into PCI Express request packets. Incoming requests can be up to 512 bytes. For better performance, Altera recommends using smaller read request size (a maximum of 512 bytes).

Table 4-3: Avalon-MM TX Slave Interface Signals

Signal Name	Direction	Description
TxsChipSelect_i	Input	The system interconnect fabric asserts this signal to select the TX slave port.
TxsRead_i	Input	Read request asserted by the system interconnect fabric to request a read.
TxsWrite_i	Input	Write request asserted by the system interconnect fabric to request a write.
TxsWriteData[127 or 63:0]	Input	Write data sent by the external Avalon-MM master to the TX slave port.
TxsBurstCount[6 or 5:0]	Input	Asserted by the system interconnect fabric indicating the amount of data requested. The count unit is the amount of data that is transferred in a single cycle, that is, the width of the bus. The burst count is limited to 512 bytes.
TxsAddress_i[<w>-1:0]	Input	Address of the read or write request from the external Avalon-MM master. This address translates to 64-bit or 32-bit PCI Express addresses based on the translation table. The <*w*> value is determined when the system is created.

Signal Name	Direction	Description

TxsByteEnable_i[<w>-1:0]	Input	Write byte enable for data. A burst must be continuous. Therefore all intermediate data phases of a burst must have a byte enable value of 0xFF. The first and final data phases of a burst can have other valid values.
		For the 128-bit interface, the following restrictions apply:
		• All bytes of a single dword must either be enabled or disabled
		• If more than 1 dword is enabled, the enabled dwords must be contiguous. The following patterns are legal:
		  • 16'bF000
		  • 16'b0F00
		  • 16'b00F0
		  • 16'b000F
		  • 16'bFF00
		  • 16'b0FF0
		  • 16'b00FF
		  • 16'bFFF0
		  • 16'b0FFF
		  • 16'bFFFF

TxsReadDataValid_o	Output	Asserted by the bridge to indicate that read data is valid.
TxsReadData_o[127 or 63:0]	Output	The bridge returns the read data on this bus when the RX read completions for the read have been received and stored in the internal buffer.
TxsWaitrequest_o	Output	Asserted by the bridge to hold off read or write data when running out of buffer space. If this signal is asserted during an operation, the master should maintain the TxsRead_i signal (or TxsWrite_i signal and TxsWriteData) stable until after TxsWaitrequest_o is deasserted. txs_Read must be deasserted when TxsWaitrequest_o is deasserted.

Clock Signals

Table 4-4: Clock Signals

Signal	Direction	Description
refclk	Input	Reference clock for the IP core. It must have the frequency specified under the **System Settings** heading in the parameter editor. This is a dedicated free running input clock to the dedicated REFCLK pin. If your design meets the following criteria: • Enables CvP • Includes an additional transceiver PHY connected to the same Transceiver Reconfiguration Controller then you must connect refclk to the mgmt_clk_clk signal of the Transceiver Reconfiguration Controller and the additional transceiver PHY. In addition, if your design includes more than one Transceiver Reconfiguration Controller on the same side of the FPGA, they all must share the mgmt_clk_clk signal.
coreclkout	Output	This is a fixed frequency clock used by the Data Link and Transaction Layers. To meet PCI Express link bandwidth constraints, this clock has minimum frequency requirements as listed in *Application Layer Clock Frequency for All Combination of Link Width, Data Rate and Application Layer Interface Width* in the *Reset and Clocks* chapter .

Related Information

Clocks on page 6-5

Reset Signals

Refer to *Reset and Clocks* for more information about the reset sequence and a block diagram of the reset logic.

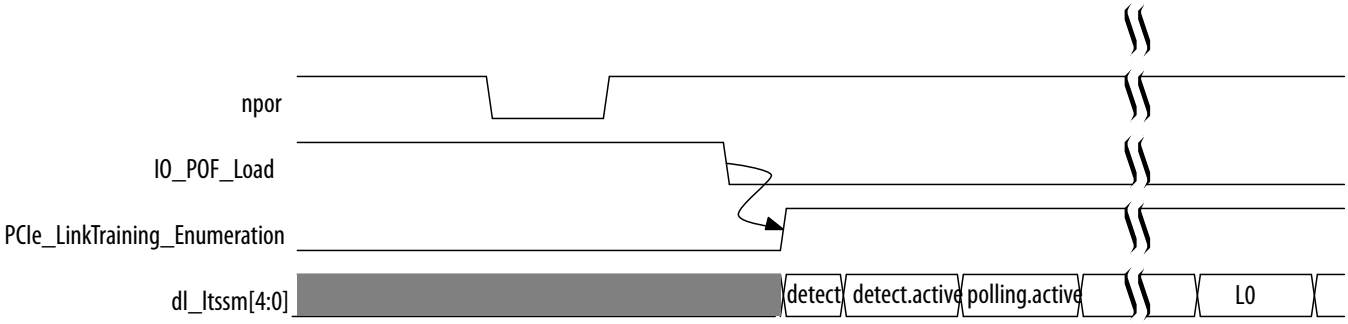
Table 4-5: Reset Signals

Signal	Direction	Description
npor	Input	Active low reset signal. In the Altera hardware example designs, npor is the OR of pin_perst and local_rstn coming from the software Application Layer. If you do not drive a soft reset signal from the Application Layer, this signal must be derived from pin_perst. You cannot disable this signal. Resets the entire IP Core and transceiver. Asynchronous. In systems that use the hard reset controller, this signal is *edge, not level* sensitive; consequently, you cannot use a low value on this signal to hold custom logic in reset. For more information about the hard and soft reset controllers, refer to the *Reset and Clocks* chapter.
nreset_status	Output	Active low reset signal. It is derived from npor or pin_perstn.
pin_perst	Input	Active low reset from the PCIe reset pin of the device. pin_perst resets the datapath and control registers. This signal is required for Configuration via Protocol (CvP). For more information about CvP refer to *Configuration via Protocol (CvP)*. Arria V have 1 or 2 instances of the Hard IP for PCI Express. Each instance has its own pin_perst signal. *You must connect the* pin_perst *of each Hard IP instance to the corresponding* nPERST *pin of the device.* These pins have the following locations: • nPERSTL0: bottom left Hard IP and CvP blocks • nPERSTL1: top left Hard IP block For example, if you are using the Hard IP instance in the bottom left corner of the device, you must connect pin_perst to nPERSL0. For maximum use of the Arria V device, Altera recommends that you use the bottom left Hard IP first. This is the only location that supports CvP over a PCIe link. Refer to the appropriate device pinout for correct pin assignment for more detailed information about these pins. The *PCI Express Card Electromechanical Specification 2.0* specifies this pin requires 3.3 V. You can drive this 3.3V signal to the nPERST*

Signal	Direction	Description
		even if the V_{CCPGM} of the bank is not 3.3V if the following 2 conditions are met: • The input signal meets the V_{IH} and V_{IL} specification for LVTTL. • The input signal meets the overshoot specification for 100°C operation as specified by the "Maximum Allowed Overshoot and Undershoot Voltage" in the *Device Datasheet for Arria V Devices*.

Figure 4-2: Reset and Link Training Timing Relationships

The following figure illustrates the timing relationship between npor and the LTSSM L0 state.



Note: To meet the 100 ms system configuration time, you must use the fast passive parallel configuration scheme with and a 32-bit data width (FPP x32).

Related Information

- **PCI Express Card Electromechanical Specification 2.0**
- **Device Datasheet for Arria V Devices**

Hard IP Status

Refer to *Reset and Clocks* for more information about the reset sequence and a block diagram of the reset logic.

Table 4-6: Status and Link Training Signals

Signal	Direction	Description
derr_cor_ext_rcv	Output	Indicates a corrected error in the RX buffer. This signal is for debug only. It is not valid until the RX buffer is filled with data. This is a pulse, not a level, signal. Internally, the pulse is generated with the 500 MHz clock. A pulse extender extends the signal so that the FPGA fabric running at 250 MHz can capture it. Because the error was corrected by the IP core, no Application Layer intervention is required.
derr_cor_ext_rpl	Output	Indicates a corrected ECC error in the retry buffer. This signal is for debug only. Because the error was corrected by the IP core, no Application Layer intervention is required.
derr_rpl	Output	Indicates an uncorrectable error in the retry buffer. This signal is for debug only. The signal is not available for Arria V and Cyclone V devices.
dlup	Output	When asserted, indicates that the Hard IP block is in the Data Link Control and Management State Machine (DLCMSM) DL_Up state.
dlup_exit	Output	This signal is asserted low for one pld_clk cycle when the IP core exits the DLCMSM DL_Up state, indicating that the Data Link Layer has lost communication with the other end of the PCIe link and left the Up state. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.
ev128ns	Output	Asserted every 128 ns to create a time base aligned activity.
ev1us	Output	Asserted every 1µs to create a time base aligned activity.
hotrst_exit	Output	Hot reset exit. This signal is asserted for 1 clock cycle when the LTSSM exits the hot reset state. This signal should cause the Application Layer to be reset. This signal is active low. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.

(3) Debug signals are not rigorously verified and should only be used to observe behavior. Debug signals should not be used to drive custom logic.

Signal	Direction	Description
int_status[3:0]	Output	These signals drive legacy interrupts to the Application Layer as follows: • int_status[0]: interrupt signal A • int_status[1]: interrupt signal B • int_status[2]: interrupt signal C • int_status[3]: interrupt signal D
ko_cpl_spc_data[11:0]	Output	The Application Layer can use this signal to build circuitry to prevent RX buffer overflow for completion data. Endpoints must advertise infinite space for completion data; however, RX buffer space is finite. ko_cpl_spc_data is a static signal that reflects the total number of 16 byte completion data units that can be stored in the completion RX buffer.
ko_cpl_spc_ header[7:0]	Output	The Application Layer can use this signal to build circuitry to prevent RX buffer overflow for completion headers. Endpoints must advertise infinite space for completion headers; however, RX buffer space is finite. ko_cpl_spc_header is a static signal that indicates the total number of completion headers that can be stored in the RX buffer.
l2_exit	Output	L2 exit. This signal is active low and otherwise remains high. It is asserted for one cycle (changing value from 1 to 0 and back to 1) after the LTSSM transitions from l2.idle to detect. When this pulse is asserted, the Application Layer should generate an internal reset signal that is asserted for at least 32 cycles.
lane_act[3:0]	Output	Lane Active Mode: This signal indicates the number of lanes that configured during link training. The following encodings are defined: • 4'b0001: 1 lane • 4'b0010: 2 lanes • 4'b0100: 4 lanes • 4'b1000: 8 lanes
ltssmstate[4:0]	Output	LTSSM state: The LTSSM state machine encoding defines the following states: • 00000: Detect.Quiet • 00001: Detect.Active • 00010: Polling.Active • 00011: Polling.Compliance • 00100: Polling.Configuration • 00101: Polling.Speed

Signal	Direction	Description
		• 00110: config.Linkwidthstart • 00111: Config.Linkaccept • 01000: Config.Lanenumaccept • 01001: Config.Lanenumwait • 01010: Config.Complete • 01011: Config.Idle • 01100: Recovery.Rcvlock • 01101: Recovery.Rcvconfig • 01110: Recovery.Idle • 01111: L0 • 10000: Disable • 10001: Loopback.Entry • 10010: Loopback.Active • 10011: Loopback.Exit • 10100: Hot.Reset • 10101: LOs • 11001: L2.transmit.Wake • 11010: Speed.Recovery • 11011: Recovery.Equalization, Phase 0 • 11100: Recovery.Equalization, Phase 1 • 11101: Recovery.Equalization, Phase 2 • 11110: recovery.Equalization, Phase 3

Related Information

PCI Express Card Electromechanical Specification 2.0

Interrupts for Endpoints when Multiple MSI/MSI-X Support Is Enabled

Table 4-7: Exported Interrupt Signals for Endpoints when Multiple MSI/MSI-X Support is Enabled

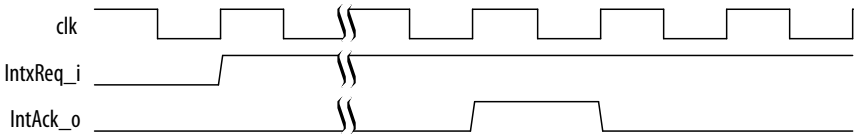
The following table describes the IP core's exported interrupt signals when you turn on **Enable multiple MSI/MSI-X support** under the **Avalon-MM System Settings** banner in the parameter editor.

Signal	Direction	Description
`MsiIntfc_o[81:0]`	Output	This bus provides the following MSI address, data, and enabled signals: • `MsiIntf_o[81]`: Master enable • `MsiIntf_o[80}`: MSI enable • `MsiIntf_o[79:64]`: MSI data • `MsiIntf_o[63:0]`: MSI address

Signal	Direction	Description
MsiControl_o[15:0]	Output	Provides for system software control of MSI as defined in Section 6.8.1.3 *Message Control for MSI* in the *PCI Local Bus Specification, Rev. 3.0*. The following fields are defined: • MsiControl_o[15:9]: Reserved • MsiControl_o[8]: Per-vector masking capable • MsiControl_o[7]: 64-bit address capable • MsiControl_o[6:4]: Multiple Message Enable • MsiControl_o[3:1]: MSI Message Capable • MsiControl_o[0]: MSI Enable.
MsixIntfc_o[15:0]	Output	Provides for system software control of MSI-X as defined in Section 6.8.2.3 *Message Control for MSI-X* in the *PCI Local Bus Specification, Rev. 3.0*. The following fields are defined: • MsixIntfc_o[15]: Enable • MsixIntfc_o[14]: Mask • MsixIntfc_o[13:11]: Reserved • MsixIntfc_o[10:0]: Table size
IntxReq_i	Input	When asserted, the Endpoint is requesting attention from the interrupt service routine unless MSI or MSI-X interrupts are enabled. Remains asserted until the device driver clears the pending request.
IntxAck_o	Output	This signal is the acknowledge for IntxReq_i. It is asserted for at least one cycle either when either of the following events occur: • The Assert_INTA message TLP has been transmitted in response to the assertion of the IntxReq_i. • The Deassert_INTA message TLP has been transmitted in response to the deassertion of the IntxReq_i signal. Refer to the timing diagrams below.

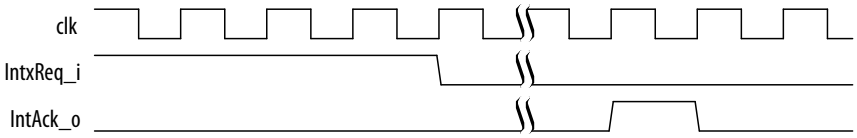
The following figure illustrates interrupt timing for the legacy interface. In this figure the assertion of IntxReq_i instructs the Hard IP for PCI Express to send an Assert_INTA message TLP.

Figure 4-3: Legacy Interrupt Assertion



The following figure illustrates the timing for deassertion of legacy interrupts. The assertion of IntxReq_i instructs the Hard IP for PCI Express to send a Deassert_INTA message.

Figure 4-4: Legacy Interrupt Deassertion



Physical Layer Interface Signals

Altera provides an integrated solution with the Transaction, Data Link and Physical Layers. The IP Parameter Editor generates a SERDES variation file, **<*variation*>_serdes.v** or **.vhd** , in addition to the Hard IP variation file, **<*variation*>.v** or **.vhd**. The SERDES entity is included in the library files for PCI Express.

Transceiver Reconfiguration

Dynamic reconfiguration compensates for variations due to process, voltage and temperature (PVT). Among the analog settings that you can reconfigure are $\mathbf{V}_{OD}$, pre-emphasis, and equalization.

You can use the Altera Transceiver Reconfiguration Controller to dynamically reconfigure analog settings. For Gen2 operation, you must turn on **Enable duty cycle calibration** in the Transceiver Reconfiguration Controller GUI. Arria V devices require duty cycle calibration (DCD) for data rates greater than 4.9152 Gbps. For more information about instantiating the Altera Transceiver Reconfiguration Controller IP core refer to *Hard IP Reconfiguration* .

Table 4-8: Transceiver Control Signals

In this table, <*n*> is the number of interfaces required.

Signal Name	Direction	Description
`reconfig_from_xcvr[(<n>46)-1:0]`	Output	Reconfiguration signals to the Transceiver Reconfiguration Controller.
`reconfig_to_xcvr[(<n>70)-1:0]`	Input	Reconfiguration signals from the Transceiver Reconfiguration Controller.
`busy_xcvr_reconfig`	Input	When asserted, indicates that the a reconfiguration operation is in progress.
`reconfig_clk_locked`	Output	When asserted, indicates that the PLL that provides the fixed clock required for transceiver initialization is locked. The Application Layer should be held in reset until `reconfig_clk_locked` is asserted.

The following table shows the number of logical reconfiguration and physical interfaces required for various configurations. The Quartus II Fitter merges logical interfaces to minimize the number of physical interfaces configured in the hardware. Typically, one logical interface is required for each channel and one for each PLL.

The ×8 variants require an extra channel for PCS clock routing and control. The ×8 variants use channel 4 for clocking.

Table 4-9: Number of Logical and Physical Reconfiguration Interfaces

Variant	Logical Interfaces
Gen1 and Gen2 ×1	2
Gen1 and Gen2 ×2	3
Gen1 and Gen2 ×4	5
Gen1 ×8	10

For more information about the Transceiver Reconfiguration Controller, refer to the *Transceiver Reconfiguration Controller* chapter in the *Altera Transceiver PHY IP Core User Guide* .

Related Information

Altera Transceiver PHY IP Core User Guide

Hard IP Status Extension

Table 4-10: Hard IP Status Extension Signals

This optional bus adds signals that are useful for debugging to the top-level variant, including:

- The most important native Avalon-ST RX signals
- The Configuration Space signals
- The BAR
- The ECC error
- The signal indicating that the `pld_clk` is in use

Signal	Direction	Description
`pld_clk_inuse`	Output	When asserted, indicates that the Hard IP Transaction Layer is using the `pld_clk` as its clock and is ready for operation with the Application Layer. For reliable operation, hold the Application Layer in reset until `pld_clk_inuse` is asserted.
`pme_to_sr`	Output	Power management turn off status register. Root Port—This signal is asserted for 1 clock cycle when the Root Port receives the `pme_turn_off` acknowledge message. Endpoint—This signal is asserted for 1 cycle when the Endpoint receives the `PME_turn_off` message from the Root Port.

Signal	Direction	Description
rx_st_bar[7:0]	Output	The decoded BAR bits for the TLP. Valid for MRd, MWr, IOWR, and IORD TLPs. Ignored for the completion or message TLPs. Valid during the cycle in which rx_st_sop is asserted. The following encodings are defined for Endpoints: • Bit 0: BAR 0 • Bit 1: BAR 1 • Bit 2: Bar 2 • Bit 3: Bar 3 • Bit 4: Bar 4 • Bit 5: Bar 5 • Bit 6: Reserved • Bit 7: Reserved The following encodings are defined for Root Ports: • Bit 0: BAR 0 • Bit 1: BAR 1 • Bit 2: Primary Bus number • Bit 3: Secondary Bus number • Bit 4: Secondary Bus number to Subordinate Bus number window • Bit 5: I/O window • Bit 6: Non-Prefetchable window • Bit 7: Prefetchable window
rx_st_data[<*n*>-1:0]	Output	Receive data bus. Note that the position of the first payload dword depends on whether the TLP address is qword aligned. The mapping of message TLPs is the same as the mapping of TLPs with 4-dword headers.
rx_st_eop	Output	Indicates that this is the last cycle of the TLP when rx_st_valid is asserted.
rx_st_err	Output	Indicates that there is an ECC error in the internal RX buffer. Active when ECC is enabled. ECC is automatically enabled by the Quartus II assembler. ECC corrects single-bit errors and detects double-bit errors on a per byte basis. When an uncorrectable ECC error is detected, rx_st_err is asserted for at least 1 cycle while rx_st_valid is asserted. Altera recommends resetting the Arria V Hard IP for PCI Express when an uncorrectable double-bit ECC error is detected.

Signal	Direction	Description
rx_st_sop	Output	Indicates that this is the first cycle of the TLP when rx_st_valid is asserted.
rx_st_valid	Output	Clocks rx_st_data into the Application Layer. Deasserts within 2 clocks of rx_st_ready deassertion and reasserts within 2 clocks of rx_st_ready assertion if more data is available to send.
serr_out	Output	System Error: This signal only applies to Root Port designs that report each system error detected, assuming the proper enabling bits are asserted in the Root Control and Device Control registers. If enabled, serr_out is asserted for a single clock cycle when a system error occurs. System errors are described in the *PCI Express Base Specification 2.1 or 3.0* in the Root Control register.
tl_cfg_add[3:0]	Output	Address of the register that has been updated. This signal is an index indicating which Configuration Space register information is being driven onto tl_cfg_ctl.
tl_cfg_sts[52:0]	Output	Configuration status bits. This information updates every pld_clk cycle. The following table provides detailed descriptions of the status bits.
tx_st_ready	Output	Indicates that the Transaction Layer is ready to accept data for transmission. The core deasserts this signal to throttle the data stream. tx_st_ready may be asserted during reset. The Application Layer should wait at least 2 clock cycles after the reset is released before issuing packets on the Avalon-ST TX interface. The reset_status signal can also be used to monitor when the IP core has come out of reset. If asserted by the Transaction Layer on cycle <*n*>*tx_st_ready*, then <*n* + readyLatency> is a ready cycle, during which the Application Layer may assert tx_st_valid and transfer data. When tx_st_ready, tx_st_valid and tx_st_data are registered (the typical case), Altera recommends a readyLatency of 2 cycles to facilitate timing closure; however, a readyLatency of 1 cycle is possible. If no other delays are added to the read-valid latency, the resulting delay corresponds to a readyLatency of 2.

Table 4-11: Mapping Between tl_cfg_sts and Configuration Space Registers

tl_cfg_sts	Configuration Space Register	Description
[52:49]	Device Status Register[3:0]	Records the following errors: • Bit 3: unsupported request detected • Bit 2: fatal error detected • Bit 1: non-fatal error detected • Bit 0: correctable error detected
[48]	Slot Status Register[8]	Data Link Layer state changed
[47]	Slot Status Register[4]	Command completed. (The hot plug controller completed a command.) **Note:** For Root Ports, you enable the Slot register by turning on **Use Slot Power Register** in the parameter editor. When enabled, access to Command Completed Interrupt Enable bit of the Slot Control register remains Read/Write. This bit should be hardwired to 1b'0. You should not write this bit.
[46:31]	Link Status Register[15:0]	Records the following link status information: • Bit 15: link autonomous bandwidth status • Bit 14: link bandwidth management status • Bit 13: Data Link Layer link active • Bit 12: Slot clock configuration • Bit 11: Link Training • Bit 10: Undefined • Bits[9:4]: Negotiated Link Width • Bits[3:0] Link Speed
[30]	Link Status 2 Register[0]	Current de-emphasis level.

tl_cfg_sts	Configuration Space Register	Description
[29:25]	Status Register[15:11]	Records the following 5 primary command status errors: • Bit 15: detected parity error • Bit 14: signaled system error • Bit 13: received master abort • Bit 12: received target abort • Bit 11: signalled target abort
[24]	Secondary Status Register[8]	Master data parity error
[23:6]	Root Status Register[17:0]	Records the following PME status information: • Bit 17: PME pending • Bit 16: PME status • Bits[15:0]: PME request ID[15:0]
[5:1]	Secondary Status Register[15:11]	Records the following 5 secondary command status errors: • Bit 15: detected parity error • Bit 14: received system error • Bit 13: received master abort • Bit 12: received target abort • Bit 11: signalled target abort
[0]	Secondary Status Register[8]	Master Data Parity Error

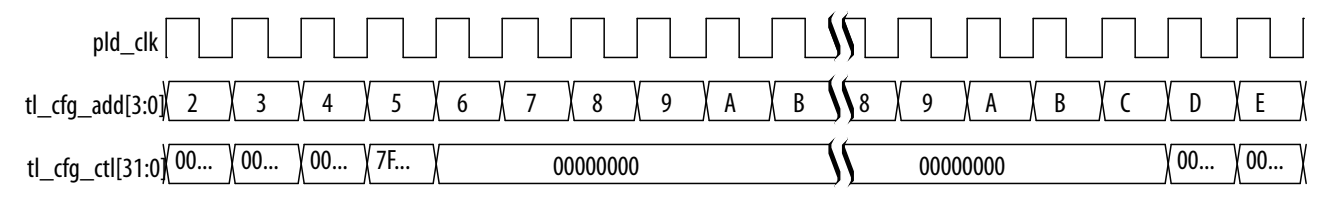
Related Information

PCI Express Card Electromechanical Specification 2.0

Configuration Space Register Access Timing

Figure 4-5: tl_cfg_ctl Timing

The following figure shows typical traffic on the `tl_cfg_ctl` bus. The `tl_cfg_add` index increments on the rising edge of the `pld_clk`. The address specifies which Configuration Space register data value is being driven onto `tl_cfg_ctl`.



Configuration Space Register Access

The `tl_cfg_ctl` signal is a multiplexed bus that contains the contents of Configuration Space registers as shown in the figure below. Information stored in the Configuration Space is accessed in round robin order where `tl_cfg_add` indicates which register is being accessed. The following table shows the layout of configuration information that is multiplexed on `tl_cfg_ctl`.

Figure 4-6: Multiplexed Configuration Register Information Available on tl_cfg_ctl

Fields in blue are available only for Root Ports.

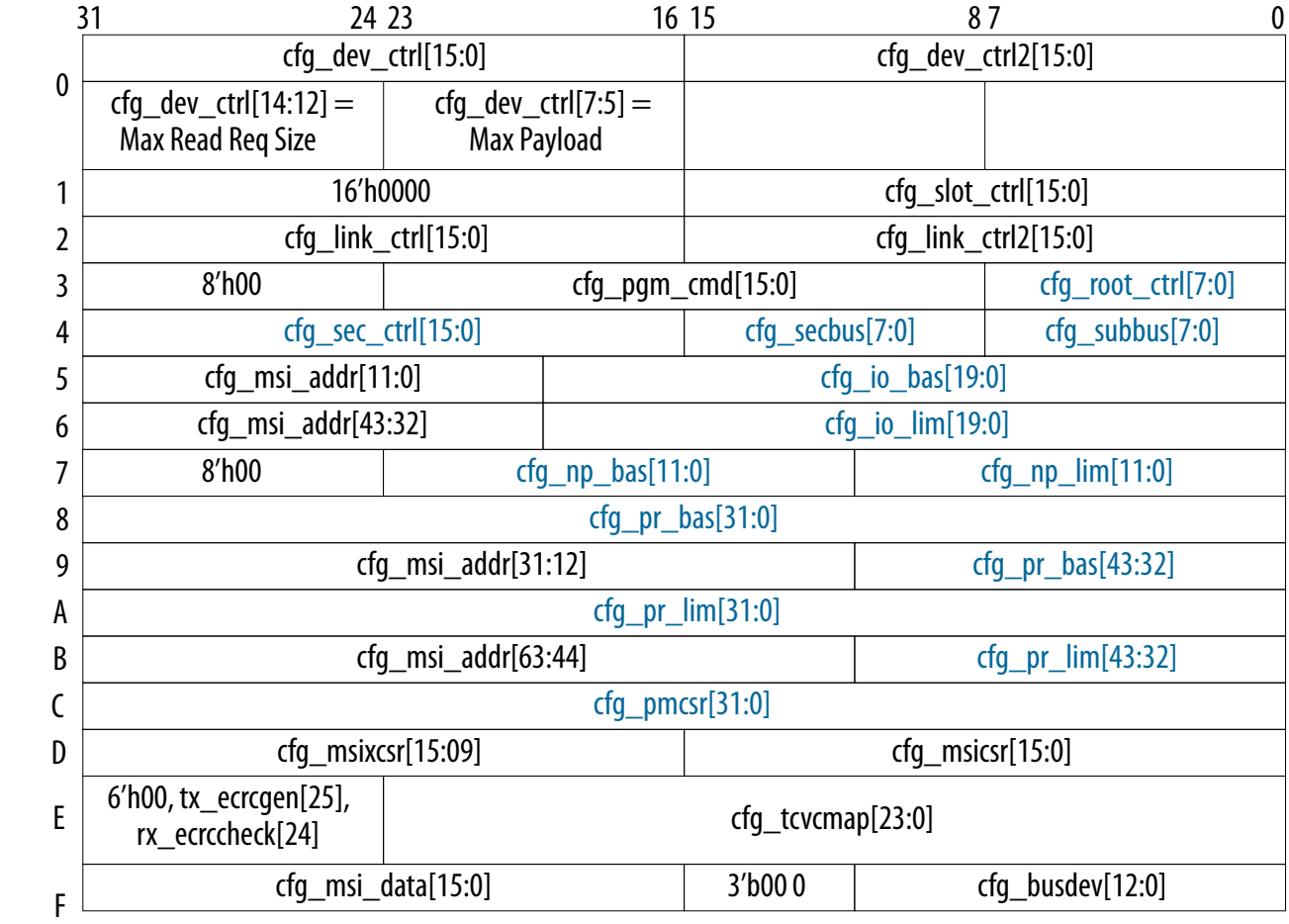


Table 4-12: Configuration Space Register Descriptions

Register	Width	Direction	Description
`cfg_dev_ctrl_func<n>`	16	Output	`cfg_dev_ctrl_func<n>[15:0]` is Device Control register for the PCI Express capability structure.
`cfg_dev_ctrl2`	16	Output	`cfg_dev2ctrl[15:0]` is Device Control 2 for the PCI Express capability structure.

Register	Width	Direction	Description
cfg_slot_ctrl	16	Output	cfg_slot_ctrl[15:0] is the Slot Status of the PCI Express capability structure. This register is only available in Root Port mode.
cfg_link_ctrl	16	Output	cfg_link_ctrl[15:0]is the primary Link Control of the PCI Express capability structure. For Gen2 operation, you must write a 1'b1 to the Retrain Link bit (Bit[5] of the cfg_link_ctrl) of the Root Port to initiate retraining to a higher data rate after the initial link training to Gen1 L0 state. Retraining directs the LTSSM to the Recovery state. Retraining to a higher data rate is not automatic for the Arria V Hard IP for PCI Express IP Core even if both devices on the link are capable of a higher data rate.
cfg_link_ctrl2	16	Output	cfg_link_ctrl2[31:16] is the secondary Link Control register of the PCI Express capability structure for Gen2 operation. When tl_cfg_addr=4'b0010, tl_cfg_ctl returns the primary and secondary Link Control registers, {{cfg_link_ctrl[15:0], cfg_link_ctrl2[15:0]}. The primary Link Status register contents are available on tl_cfg_sts[46:31]. For Gen1 variants, the link bandwidth notification bit is always set to 0. For Gen2 variants, this bit is set to 1.
cfg_prm_cmd_func<*n*>	16	Output	Base/Primary Command register for the PCI Configuration Space.
cfg_root_ctrl	8	Output	Root control and status register of the PCI Express capability. This register is only available in Root Port mode.
cfg_sec_ctrl	16	Output	Secondary bus Control and Status register of the PCI Express capability. This register is available only in Root Port mode.
cfg_secbus	8	Output	Secondary bus number. This register is available only in Root Port mode.
cfg_subbus	8	Output	Subordinate bus number. This register is available only in Root Port mode.

Register	Width	Direction	Description
cfg_msi_addr	64	Output	cfg_msi_add[63:32] is the message signaled interrupt (MSI) upper message address. cfg_msi_add[31:0] is the MSI message address.
cfg_io_bas	20	Output	The upper 20 bits of the I/O limit registers of the Type1 Configuration Space. This register is only available in Root Port mode.
cfg_io_lim	20	Output	The upper 20 bits of the IO limit registers of the Type1 Configuration Space. This register is only available in Root Port mode.
cfg_np_bas	12	Output	The upper 12 bits of the memory base register of the Type1 Configuration Space. This register is only available in Root Port mode.
cfg_np_lim	12	Output	The upper 12 bits of the memory limit register of the Type1 Configuration Space. This register is only available in Root Port mode.
cfg_pr_bas	44	Output	The upper 44 bits of the prefetchable base registers of the Type1 Configuration Space. This register is only available in Root Port mode.
cfg_pr_lim	44	Output	The upper 44 bits of the prefetchable limit registers of the Type1 Configuration Space. Available in Root Port mode.
cfg_pmcsr	32	Output	cfg_pmcsr[31:16] is Power Management Control and cfg_pmcsr[15:0]is the Power Management Status register.
cfg_msixcsr	16	Output	MSI-X message control.
cfg_msicsr	16	Output	MSI message control. Refer to the following table for the fields of this register.

Register	Width	Direction	Description
cfg_tcvcmap	24	Output	Configuration traffic class (TC)/virtual channel (VC) mapping. The Application Layer uses this signal to generate a TLP mapped to the appropriate channel based on the traffic class of the packet. • cfg_tcvcmap[2:0]: Mapping for TC0 (always 0) . • cfg_tcvcmap[5:3]: Mapping for TC1. • cfg_tcvcmap[8:6]: Mapping for TC2. • cfg_tcvcmap[11:9]: Mapping for TC3. • cfg_tcvcmap[14:12]: Mapping for TC4. • cfg_tcvcmap[17:15]: Mapping for TC5. • cfg_tcvcmap[20:18]: Mapping for TC6. • cfg_tcvcmap[23:21]: Mapping for TC7.
cfg_msi_data	16	Output	cfg_msi_data[15:0] is message data for MSI.
cfg_busdev	13	Output	Bus/Device Number captured by or programmed in the Hard IP.

Figure 4-7: Configuration MSI Control Status Register

Field and Bit Map					
15 9	8	7	6 4	3 1	0
reserved	mask capability	64-bit address capability	multiple message enable	multiple message capable	MSI enable

Table 4-13: Configuration MSI Control Status Register Field Descriptions

Bit(s)	Field	Description
[15:9]	Reserved	N/A
[8]	mask capability	Per-vector masking capable. This bit is hardwired to 0 because the function does not support the optional MSI per-vector masking using the Mask_Bits and Pending_Bits registers defined in the *PCI Local Bus Specification*. Per-vector masking can be implemented using Application Layer registers.
[7]	64-bit address capability	64-bit address capable. • 1: function capable of sending a 64-bit message address • 0: function not capable of sending a 64-bit message address

Bit(s)	Field	Description
[6:4]	multiple message enable	This field indicates permitted values for MSI signals. For example, if "100" is written to this field 16 MSI signals are allocated. • 3'b000: 1 MSI allocated • 3'b001: 2 MSI allocated • 3'b010: 4 MSI allocated • 3'b011: 8 MSI allocated • 3'b100: 16 MSI allocated • 3'b101: 32 MSI allocated • 3'b110: Reserved • 3'b111: Reserved
[3:1]	multiple message capable	This field is read by system software to determine the number of requested MSI messages. • 3'b000: 1 MSI requested • 3'b001: 2 MSI requested • 3'b010: 4 MSI requested • 3'b011: 8 MSI requested • 3'b100: 16 MSI requested • 3'b101: 32 MSI requested • 3'b110: Reserved
[0]	MSI Enable	If set to 0, this component is not permitted to use MSI.

Related Information

- **PCI Express Base Specification 2.1 or 3.0**
- **PCI Local Bus Specification, Rev. 3.0**

Serial Interface Signals

Table 4-14: Serial Interface Signals

In the following table, <*n*> = 1, 2, 4, or 8.

Signal	Direction	Description
tx_out[<*n*>-1:0]	Output	Transmit input. These signals are the serial outputs.
rx_in[<*n*>-1:0]	Input	Receive input. These signals are the serial inputs.

Refer to *Pin-out Files for Altera Devices* for pin-out tables for all Altera devices in **.pdf**, **.txt**, and **.xls** formats.

Related Information

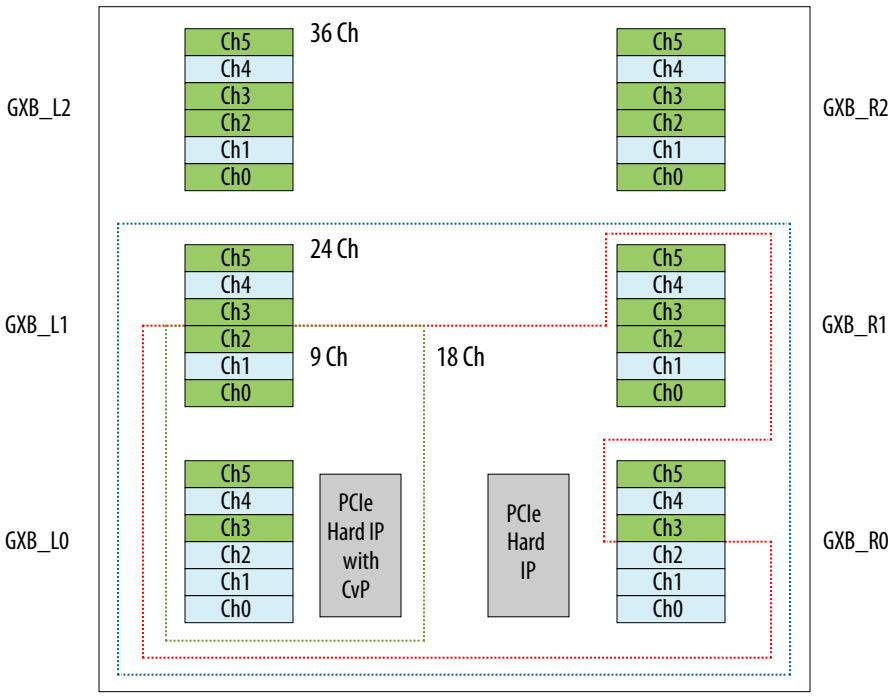
Pin-out Files for Altera Devices

Physical Layout of Hard IP in Arria V Devices

/>Arria V devices include one or two Hard IP for PCI Express IP cores. The following figures illustrate the placement of the PCIe IP cores, transceiver banks, and channels. Note that the bottom left IP core includes the CvP functionality. The other Hard IP blocks do not include the CvP functionality.

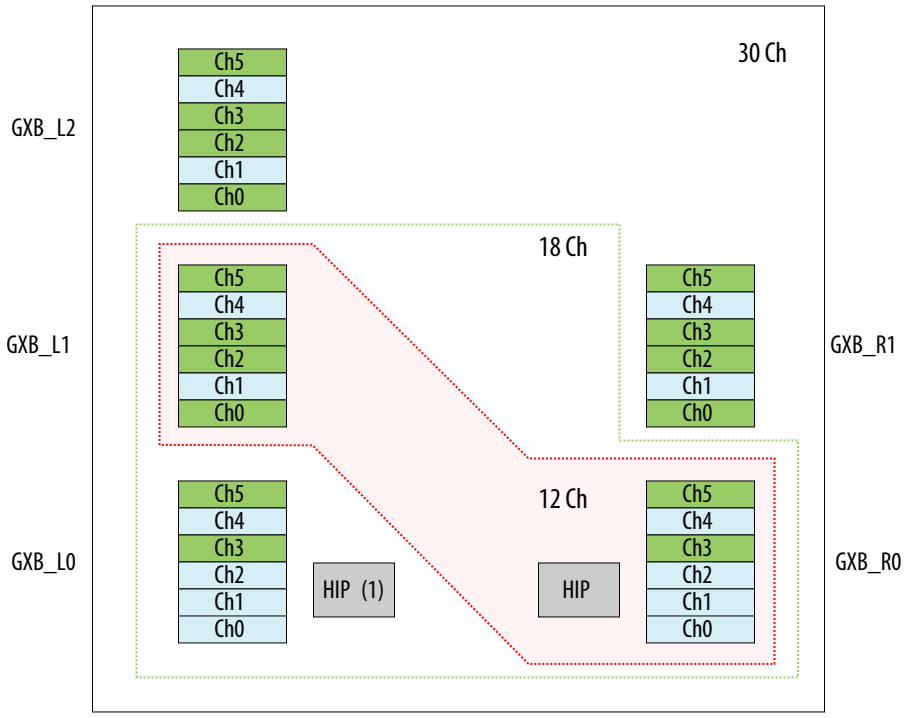
Transceiver channels are arranged in groups of six. For GX devices, the lowest six channels on the left side of the device are labeled GXB_L0, the next group is GXB_L1, and so on. Channels on the right side of the device are labeled GXB_R0, GXB_R1, and so on. Be sure to connect the Hard IP for PCI Express on the left side of the device to appropriate channels on the left side of the device, as specified in the *Pin-out Files for Altera Devices*.

Figure 4-8: Arria V Transceiver Bank and Hard IP for PCI Express IP Core Locations in Arria V GX and GT Devices



Notes:
1. Green blocks are 10-Gbps channels.
2. Blue blocks are 6-Gbps channels.

Figure 4-9: Arria V Transceiver Bank and Hard IP for PCI Express IP Core Locations in Arria V SX and ST Devices



Notes:
1. PCIe HIP availability varies with device variants.
2. Green blocks are 10-Gbps channels.
3. Blue blocks are 6-Gbps channels. With the exception of Ch0 to Ch2 in GXB_L0 and GXB_R0, the 6-Gbps channels can be used for TX-only or RX-only 10-Gbps channels.

Channel utilization for x1, x2, x4, and x8 variants is as follows:

Table 4-15: Channel Utilization

Variant	Data	CMU Clock
x1, 1 instance	Channel 0 of GXB_L0	Channel 1 of GXB_L0
x1, 2 instances	Channel 0 of GXB_L0, Channel 0 of GXB_R0	Channel 1 of GXB_L0, Channel 1 of GXB_R0
x2, 1 instance	Channels 1–2 of GXB_L0	Channel 4 of GXB_L0
x2, 2 instances	Channels 1–2 of GXB_L0, Channels 1–2 of GXB_R0	Channel 4 of GXB_L0, Channel 4 of GXB_R0
x4, 1 instance	Channels 0–3 of GXB_L0	Channel 4 of GXB_L0
x4, 2 instances	Channels 0–3 of GXB_L0, Channels 0–3 of GXB_R0	Channel 4 of GXB_L0, Channel 4 of GXB_R0
x8, 1 instance	Channels 0–3 and 5 of GXB_L0 and channels 0-2 of GXB_L1	Channel 4 of GXB_L0

For more comprehensive information about Arria V transceivers, refer to the *Transceiver Banks* section in the *Transceiver Architecture in Arria V Devices*.

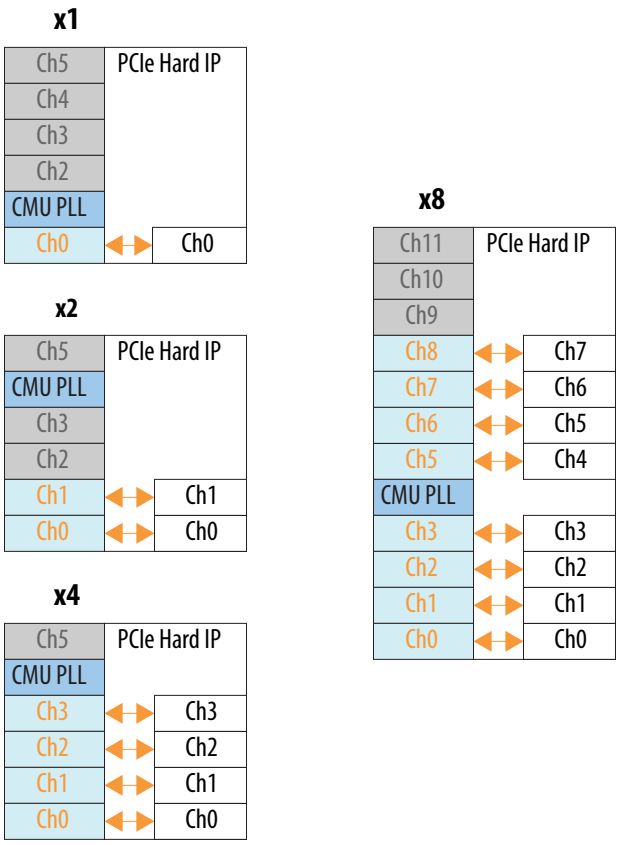
Related Information

Transceiver Architecture in Arria V Devices

Channel Placement in Arria V Devices

Figure 4-10: Arria V Gen1 and Gen2 Channel Placement Using the CMU PLL

In the following figures the channels shaded in blue provide the transmit CMU PLL generating the high-speed serial clock.



You can assign other protocols to unused channels the if data rate and clock specification exactly match the PCIe configuration.

PIPE Interface Signals

These PIPE signals are available for Gen1 and Gen2 variants so that you can simulate using either the serial or the PIPE interface. Simulation is much faster using the PIPE interface because the PIPE simulation bypasses the SERDES model . By default, the PIPE interface is 8 bits for Gen1 and Gen2. You can use the PIPE interface for simulation even though your actual design includes a serial interface to the internal transceivers. However, it is not possible to use the Hard IP PIPE interface in hardware, including probing these signals using SignalTap® II Embedded Logic Analyzer.

Table 4-16: PIPE Interface Signals

In the following table, signals that include lane number 0 also exist for other lanes.

Signal	Direction	Description
txdata0[7:0]	Output	Transmit data <*n*> (2 symbols on lane <*n*>). This bus transmits data on lane <*n*>.
txdatak0	Output	Transmit data control <*n*>. This signal serves as the control bit for txdata <*n*>.
txdetectrx0	Output	Transmit detect receive <*n*>. This signal tells the PHY layer to start a receive detection operation or to begin loopback.
txelecidle0	Output	Transmit electrical idle <*n*>. This signal forces the TX output to electrical idle.
txcompl0	Output	Transmit compliance <*n*>. This signal forces the running disparity to negative in Compliance Mode (negative COM character).
rxpolarity0	Output	Receive polarity <*n*>. This signal instructs the PHY layer to invert the polarity of the 8B/10B receiver decoding block.
powerdown0[1:0]	Output	Power down <*n*>. This signal requests the PHY to change its power state to the specified state (P0, P0s, P1, or P2).
tx_deemph0	Output	Transmit de-emphasis selection. The Arria V Hard IP for PCI Express sets the value for this signal based on the indication received from the other end of the link during the Training Sequences (TS). You do not need to change this value.
rxdata0[7:0] (1)	Input	Receive data <*n*> (2 symbols on lane <*n*>). This bus receives data on lane <*n*>.
rxdatak0 (1)	Input	Receive data >*n*>. This bus receives data on lane <*n*>.
rxvalid0 (1)	Input	Receive valid <*n*>. This signal indicates symbol lock and valid data on rxdata<*n*> and rxdatak <*n*>.
phystatus0 (1)	Input	PHY status <*n*>. This signal communicates completion of several PHY requests.

Signal	Direction	Description
eidleinfersel0[2:0]	Output	Electrical idle entry inference mechanism selection. The following encodings are defined: • 3'b0xx: Electrical Idle Inference not required in current LTSSM state • 3'b100: Absence of COM/SKP Ordered Set in the 128 us window for Gen1 or Gen2 • 3'b101: Absence of TS1/TS2 Ordered Set in a 1280 UI interval for Gen1 or Gen2 • 3'b110: Absence of Electrical Idle Exit in 2000 UI interval for Gen1 and 16000 UI interval for Gen2 • 3'b111: Absence of Electrical idle exit in 128 us window for Gen1
rxelecidle0 (1)	Input	Receive electrical idle <*n*>. When asserted, indicates detection of an electrical idle.
rxstatus0[2:0] (1)	Input	Receive status <*n*>. This signal encodes receive status and error codes for the receive data stream and receiver detection.
sim_pipe_ltssmstate0[4:0]	Input and Output	LTSSM state: The LTSSM state machine encoding defines the following states: • 5'b00000: Detect.Quiet • 5'b 00001: Detect.Active • 5'b00010: Polling.Active • 5'b 00011: Polling.Compliance • 5'b 00100: Polling.Configuration • 5'b00101: Polling.Speed • 5'b00110: config.LinkwidthsStart • 5'b 00111: Config.Linkaccept • 5'b 01000: Config.Lanenumaccept • 5'b01001: Config.Lanenumwait • 5'b01010: Config.Complete • 5'b 01011: Config.Idle • 5'b01100: Recovery.Rcvlock • 5'b01101: Recovery.Rcvconfig • 5'b01110: Recovery.Idle • 5'b 01111: L0 • 5'b10000: Disable • 5'b10001: Loopback.Entry • 5'b10010: Loopback.Active • 5'b10011: Loopback.Exit • 5'b10100: Hot.Reset

Signal	Direction	Description
		• 5'b10101: LOs • 5'b11001: L2.transmit.Wake • 5'b11010: Speed.Recovery • 5'b11011: Recovery.Equalization, Phase 0 • 5'b11100: Recovery.Equalization, Phase 1 • 5'b11101: Recovery.Equalization, Phase 2 • 5'b11110: Recovery.Equalization, Phase 3 • 5'b11111: Recovery.Equalization, Done
`sim_pipe_rate[1:0]`	Output	The 2-bit encodings have the following meanings: • 2'b00: Gen1 rate (2.5 Gbps) • 2'b01: Gen2 rate (5.0 Gbps) • 2'b1X: Gen3 rate (8.0 Gbps)
`sim_pipe_pclk_in`	Input	This clock is used for PIPE simulation only, and is derived from the `refclk`. It is the PIPE interface clock used for PIPE mode simulation.
`txswing0`	Output	When asserted, indicates full swing for the transmitter voltage. When deasserted indicates half swing.
`tx_margin0[2:0]`	Output	Transmit V_{OD} margin selection. The value for this signal is based on the value from the `Link Control 2 Register`. Available for simulation only.

Notes:

1. These signals are for simulation only. For Quartus II software compilation, these pipe signals can be left floating.

Test Signals

Table 4-17: Test Interface Signals

The `test_in` bus provides run-time control and monitoring of the internal state of the IP core.

Signal	Direction	Description
`test_in[31:0]`	Input	The bits of the `test_in` bus have the following definitions: • [0]: Simulation mode. This signal can be set to 1 to accelerate initialization by reducing the value of many initialization counters. • [1]: Reserved. Must be set to 1'b0. • [2]: Descramble mode disable. This signal must be set to 1 during initialization in order to disable data scrambling. You can use this bit in simulation for both Endpoints and Root Ports to observe descrambled data on the link. Descrambled data cannot be used in open systems because the link partner typically scrambles the data. • [4:3]: Reserved. Must be set to 4'b01. • [5]: Compliance test mode. Disable/force compliance mode. When set, prevents the LTSSM from entering compliance mode. Toggling this bit controls the entry and exit from the compliance state, enabling the transmission of compliance patterns. • [6]: Forces entry to compliance mode when a timeout is reached in the polling.active state and not all lanes have detected their exit condition. • [7]: Disable low power state negotiation. Altera recommends setting thist bit. • [31:8] Reserved. Set to all 0s.
`simu_mode_pipe`	Input	When high, indicates that the PIPE interface is in simulation mode.
`hip_currentspeed[1:0]`	Output	Indicates the current speed of the PCIe link. The following encodings are defined: • 2b'00: Undefined • 2b'01: Gen1 • 2b'10: Gen2 • 2b'11: Gen3

Registers 5

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

Correspondence between Configuration Space Registers and the PCIe Specification

Table 5-1: Correspondence between Configuration Space Capability Structures and PCIe Base Specification Description

For the Type 0 and Type 1 Configuration Space Headers, the first line of each entry lists Type 0 values and the second line lists Type 1 values when the values differ.

Byte Address	Hard IP Configuration Space Register	Corresponding Section in PCIe Specification
0x000:0x03C	PCI Header Type 0 Configuration Registers	Type 0 Configuration Space Header
0x000:0x03C	PCI Header Type 1 Configuration Registers	Type 1 Configuration Space Header
0x040:0x04C	Reserved	N/A
0x050:0x05C	MSI Capability Structure	MSI Capability Structure
0x068:0x070	MSI-X Capability Structure	MSI-X Capability Structure
0x070:0x074	Reserved	N/A
0x078:0x07C	Power Management Capability Structure	PCI Power Management Capability Structure
0x080:0x0B8	PCI Express Capability Structure	PCI Express Capability Structure
0x0B8:0x0FC	Reserved	N/A
0x094:0x0FF	Root Port	N/A
0x100:0x16C	Virtual Channel Capability Structure (Reserved)	Virtual Channel Capability

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Byte Address	Hard IP Configuration Space Register	Corresponding Section in PCIe Specification
0x170:0x17C	Reserved	N/A
0x180:0x1FC	Virtual channel arbitration table (Reserved)	VC Arbitration Table
0x200:0x23C	Port VC0 arbitration table (Reserved)	Port Arbitration Table
0x240:0x27C	Port VC1 arbitration table (Reserved)	Port Arbitration Table
0x280:0x2BC	Port VC2 arbitration table (Reserved)	Port Arbitration Table
0x2C0:0x2FC	Port VC3 arbitration table (Reserved)	Port Arbitration Table
0x300:0x33C	Port VC4 arbitration table (Reserved)	Port Arbitration Table
0x340:0x37C	Port VC5 arbitration table (Reserved)	Port Arbitration Table
0x380:0x3BC	Port VC6 arbitration table (Reserved)	Port Arbitration Table
0x3C0:0x3FC	Port VC7 arbitration table (Reserved)	Port Arbitration Table
0x400:0x7FC	Reserved	PCIe spec corresponding section name
0x800:0x834	Advanced Error Reporting AER (optional)	Advanced Error Reporting Capability
0x838:0xFFF	Reserved	N/A
0x000	Device ID, Vendor ID	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x004	Status, Command	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x008	Class Code, Revision ID	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x00C	BIST, Header Type, Primary Latency Timer, Cache Line Size	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x010	Base Address 0	Base Address Registers
0x014	Base Address 1	Base Address Registers

Byte Address	Hard IP Configuration Space Register	Corresponding Section in PCIe Specification
0x018	Base Address 2 Secondary Latency Timer, Subordinate Bus Number, Secondary Bus Number, Primary Bus Number	Base Address Registers Secondary Latency Timer, Type 1 Configuration Space Header, Primary Bus Number
0x01C	Base Address 3 Secondary Status, I/O Limit, I/O Base	Base Address Registers Secondary Status Register ,Type 1 Configuration Space Header
0x020	Base Address 4 Memory Limit, Memory Base	Base Address Registers Type 1 Configuration Space Header
0x024	Base Address 5 Prefetchable Memory Limit, Prefetchable Memory Base	Base Address Registers Prefetchable Memory Limit, Prefetchable Memory Base
0x028	Reserved Prefetchable Base Upper 32 Bits	N/A Type 1 Configuration Space Header
0x02C	Subsystem ID, Subsystem Vendor ID Prefetchable Limit Upper 32 Bits	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x030	I/O Limit Upper 16 Bits, I/O Base Upper 16 Bits	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x034	Reserved, Capabilities PTR	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x038	Reserved	N/A
0x03C	Interrupt Pin, Interrupt Line Bridge Control, Interrupt Pin, Interrupt Line	Type 0 Configuration Space Header Type 1 Configuration Space Header
0x050	MSI-Message Control Next Cap Ptr Capability ID	MSI and MSI-X Capability Structures
0x054	Message Address	MSI and MSI-X Capability Structures
0x058	Message Upper Address	MSI and MSI-X Capability Structures
0x05C	Reserved Message Data	MSI and MSI-X Capability Structures

Byte Address	Hard IP Configuration Space Register	Corresponding Section in PCIe Specification
0x068	MSI-X Message Control Next Cap Ptr Capability ID	MSI and MSI-X Capability Structures
0x06C	MSI-X Table Offset BIR	MSI and MSI-X Capability Structures
0x070	Pending Bit Array (PBA) Offset BIR	MSI and MSI-X Capability Structures
0x078	Capabilities Register Next Cap PTR Cap ID	PCI Power Management Capability Structure
0x07C	Data PM Control/Status Bridge Extensions Power Management Status & Control	PCI Power Management Capability Structure
0x800	PCI Express Enhanced Capability Header	Advanced Error Reporting Enhanced Capability Header
0x804	Uncorrectable Error Status Register	Uncorrectable Error Status Register
0x808	Uncorrectable Error Mask Register	Uncorrectable Error Mask Register
0x80C	Uncorrectable Error Severity Register	Uncorrectable Error Severity Register
0x810	Correctable Error Status Register	Correctable Error Status Register
0x814	Correctable Error Mask Register	Correctable Error Mask Register
0x818	Advanced Error Capabilities and Control Register	Advanced Error Capabilities and Control Register
0x81C	Header Log Register	Header Log Register
0x82C	Root Error Command	Root Error Command Register
0x830	Root Error Status	Root Error Status Register
0x834	Error Source Identification Register Correctable Error Source ID Register	Error Source Identification Register

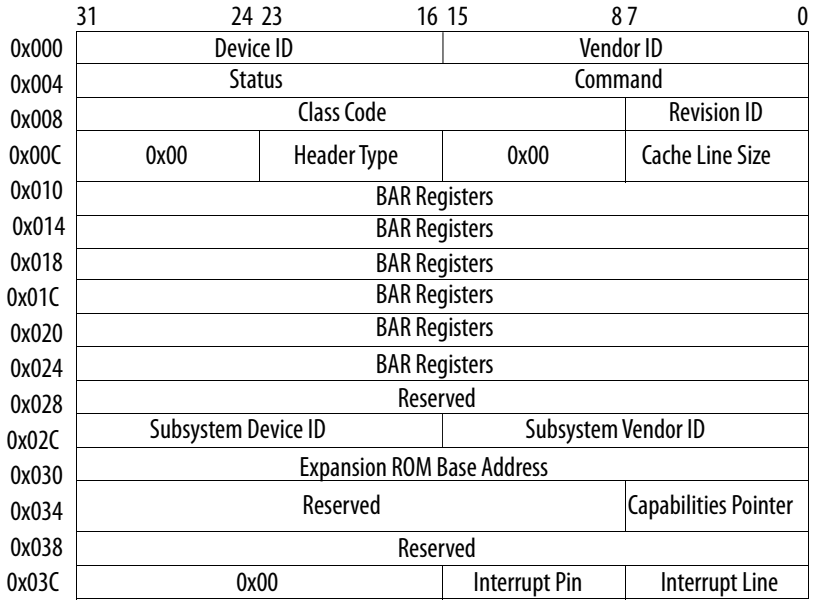
Related Information

PCI Express Base Specification 2.1 or 3.0

Type 0 Configuration Space Registers

Figure 5-1: Type 0 Configuration Space Registers - Byte Address Offsets and Layout

Endpoints store configuration data in the Type 0 Configuration Space. The **Correspondence between Configuration Space Registers and the PCIe Specification** on page 5-1 lists the appropriate section of the *PCI Express Base Specification* that describes these registers.



Type 1 Configuration Space Registers

Figure 5-2: Type 1 Configuration Space Registers (Root Ports)

	31 24	23 16	15 8	7 0
0x0000	Device ID		Vendor ID	
0x004	Status		Command	
0x008	Class Code			Revision ID
0x00C	BIST	Header Type	Primary Latency Timer	Cache Line Size
0x010	BAR Registers			
0x014	BAR Registers			
0x018	Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number
0x01C	Secondary Status		I/O Limit	I/O Base
0x020	Memory Limit		Memory Base	
0x024	Prefetchable Memory Limit		Prefetchable Memory Base	
0x028	Prefetchable Base Upper 32 Bits			
0x02C	Prefetchable Limit Upper 32 Bits			
0x030	I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits	
0x034	Reserved			Capabilities Pointer
0x038	Expansion ROM Base Address			
0x03C	Bridge Control		Interrupt Pin	Interrupt Line

PCI Express Capability Structures

Figure 5-3: MSI Capability Structure

	31 24	23 16	15 8	7 0
0x050	Message Control Configuration MSI Control Status Register Field Descriptions		Next Cap Ptr	Capability ID
0x054	Message Address			
0x058	Message Upper Address			
0x05C	Reserved		Message Data	

Figure 5-4: MSI-X Capability Structure

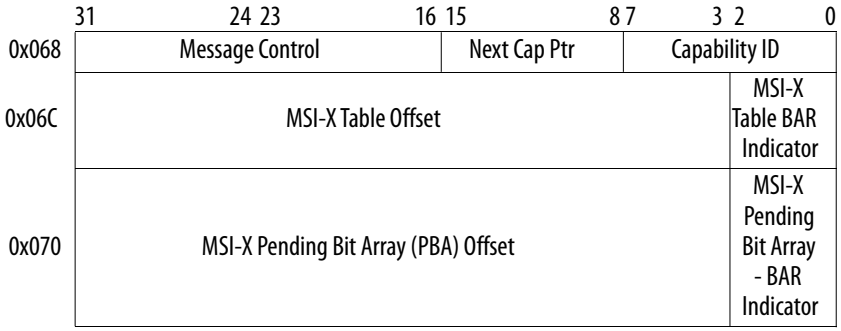


Figure 5-5: Power Management Capability Structure - Byte Address Offsets and Layout

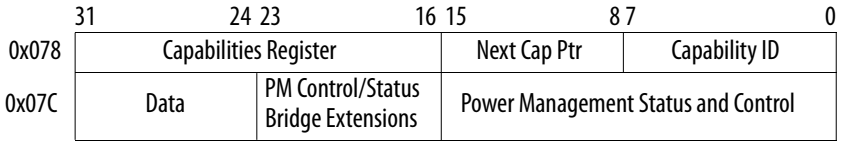
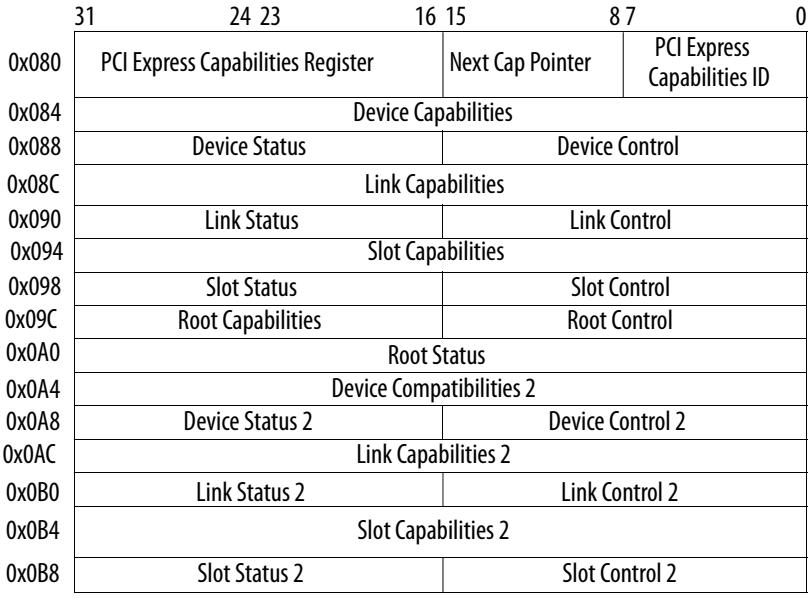


Figure 5-6: PCI Express AER Extended Capability Structure

Byte Offset	31:24	23:16	15:8	7:0
0x800	PCI Express Enhanced Capability Register			
0x804	Uncorrectable Error Status Register			
0x808	Uncorrectable Error Mask Register			
0x80C	Uncorrectable Error Severity Register			
0x810	Correctable Error Status Register			
0x814	Correctable Error Mask Register			
0x818	Advanced Error Capabilities and Control Register			
0x81C	Header Log Register			
0x82C	Root Error Command Register			
0x830	Root Error Status Register			
0x834	Error Source Identification Register		Correctable Error Source Identification Register	

Figure 5-7: PCI Express Capability Structure - Byte Address Offsets and Layout

In the following table showing the PCI Express Capability Structure, registers that are not applicable to a device are reserved.



Altera-Defined VSEC Registers

Figure 5-8: VSEC Registers

This extended capability structure supports Configuration via Protocol (CvP) programming and detailed internal error reporting.

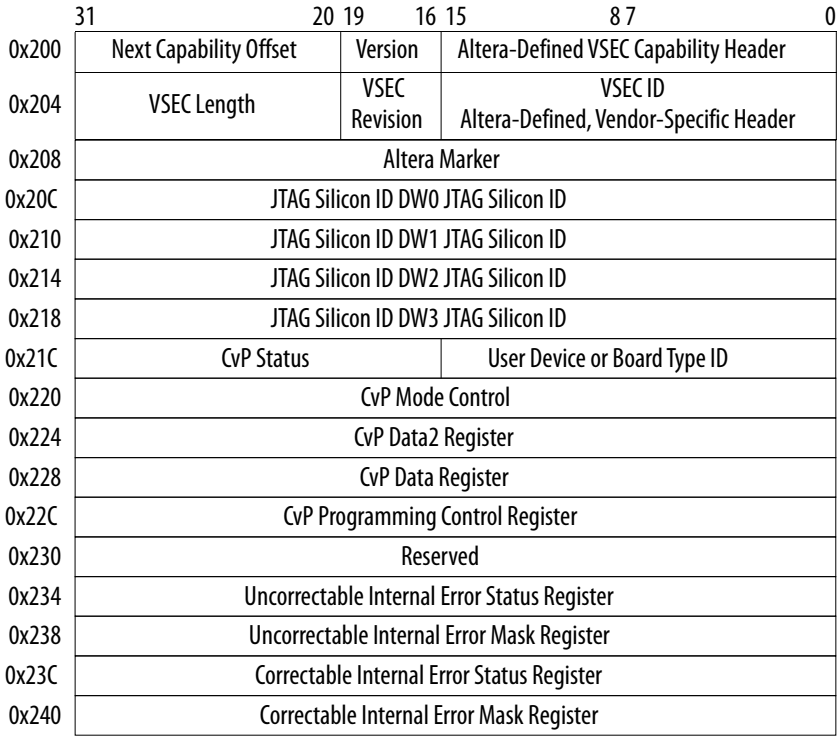


Table 5-2: Altera-Defined VSEC Capability Register, 0x200

The Altera-Defined Vendor Specific Extended Capability. This extended capability structure supports Configuration via Protocol (CvP) programming and detailed internal error reporting.

Bits	Register Description	Value	Access
[15:0]	PCI Express Extended Capability ID. Altera-defined value for VSEC Capability ID.	0x000B	RO
[19:16]	Version. Altera-defined value for VSEC version.	0x1	RO
[31:20]	Next Capability Offset. Starting address of the next Capability Structure implemented, if any.	Variable	RO

Table 5-3: Altera-Defined Vendor Specific Header

You can specify these values when you instantiate the Hard IP. These registers are read-only at run-time.

Bits	Register Description	Value	Access
[15:0]	`VSEC ID`. A user configurable VSEC ID.	User entered	RO
[19:16]	`VSEC Revision`. A user configurable VSEC revision.	Variable	RO
[31:20]	`VSEC Length`. Total length of this structure in bytes.	0x044	RO

Table 5-4: Altera Marker Register

Bits	Register Description	Value	Access
[31:0]	`Altera Marker`. This read only register is an additional marker. If you use the standard Altera Programmer software to configure the device with CvP, this marker provides a value that the programming software reads to ensure that it is operating with the correct VSEC.	A Device Value	RO

Table 5-5: JTAG Silicon ID Register

Bits	Register Description	Value	Access
[127:96]	`JTAG Silicon ID DW3`	Application Specific	RO
[95:64]	`JTAG Silicon ID DW2`	Application Specific	RO
[63:32]	`JTAG Silicon ID DW1`	Application Specific	RO
[31:0]	`JTAG Silicon ID DW0`. This is the JTAG Silicon ID that CvP programming software reads to determine that the correct SRAM object file (**.sof**) is being used.	Application Specific	RO

Table 5-6: User Device or Board Type ID Register

Bits	Register Description	Value	Access
[15:0]	Configurable device or board type ID to specify to CvP the correct **.sof**.	Variable	RO

CvP Registers

Table 5-7: CvP Status

The `CvP Status` register allows software to monitor the CvP status signals.

Bits	Register Description	Reset Value	Access
[31:26]	Reserved	0x00	RO
[25]	`PLD_CORE_READY`. From FPGA fabric. This status bit is provided for debug.	Variable	RO
[24]	`PLD_CLK_IN_USE`. From clock switch module to fabric. This status bit is provided for debug.	Variable	RO
[23]	`CVP_CONFIG_DONE`. Indicates that the FPGA control block has completed the device configuration via CvP and there were no errors.	Variable	RO
[22]	Reserved	Variable	RO
[21]	`USERMODE`. Indicates if the configurable FPGA fabric is in user mode.	Variable	RO
[20]	`CVP_EN`. Indicates if the FPGA control block has enabled CvP mode.	Variable	RO
[19]	`CVP_CONFIG_ERROR`. Reflects the value of this signal from the FPGA control block, checked by software to determine if there was an error during configuration.	Variable	RO
[18]	`CVP_CONFIG_READY`. Reflects the value of this signal from the FPGA control block, checked by software during programming algorithm.	Variable	RO
[17:0]	Reserved	Variable	RO

Table 5-8: CvP Mode Control

The `CvP Mode Control` register provides global control of the CvP operation.

Bits	Register Description	Reset Value	Access
[31:16]	Reserved.	0x0000	RO
[15:8]	`CVP_NUMCLKS`. This is the number of clocks to send for every CvP data write. Set this field to one of the values below depending on your configuration image: • 0x01 for uncompressed and unencrypted images • 0x04 for uncompressed and encrypted images • 0x08 for all compressed images	0x00	RW
[7:3]	Reserved.	0x0	RO
[2]	`CVP_FULLCONFIG`. Request that the FPGA control block reconfigure the entire FPGA including the Arria V Hard IP for PCI Express, bring the PCIe link down.	1'b0	RW

Bits	Register Description	Reset Value	Access
[1]	HIP_CLK_SEL. Selects between PMA and fabric clock when USER_MODE = 1 and PLD_CORE_READY = 1. The following encodings are defined: • 1: Selects internal clock from PMA which is required for CVP_MODE. • 0: Selects the clock from soft logic fabric. This setting should only be used when the fabric is configured in USER_MODE with a configuration file that connects the correct clock. To ensure that there is no clock switching during CvP, you should only change this value when the Hard IP for PCI Express has been idle for 10 µs and wait 10 µs after changing this value before resuming activity.	1'b0	RW
[0]	CVP_MODE. Controls whether the IP core is in CVP_MODE or normal mode. The following encodings are defined: • 1:CVP_MODE is active. Signals to the FPGA control block active and all TLPs are routed to the Configuration Space. This CVP_MODE cannot be enabled if CVP_EN = 0. • 0: The IP core is in normal mode and TLPs are routed to the FPGA fabric.	1'b0	RW

Table 5-9: CvP Data Registers

The following table defines the CvP Data registers. For 64-bit data, the optional CvP Data2 stores the upper 32 bits of data. Programming software should write the configuration data to these registers. If you Every write to these register sets the data output to the FPGA control block and generates <*n*> clock cycles to the FPGA control block as specified by the CVP_NUM_CLKS field in the CvP Mode Control register. Software must ensure that all bytes in the memory write dword are enabled. You can access this register using configuration writes, alternatively, when in CvP mode, these registers can also be written by a memory write to any address defined by a memory space BAR for this device. Using memory writes should allow for higher throughput than configuration writes.

Bits	Register Description	Reset Value	Access
[31:0]	Upper 32 bits of configuration data to be transferred to the FPGA control block to configure the device. You can choose 32- or 64-bit data.	0x00000000	RW
[31:0]	Lower 32 bits of configuration data to be transferred to the FPGA control block to configure the device.	0x00000000	RW

Table 5-10: CvP Programming Control Register

This register is written by the programming software to control CvP programming.

Bits	Register Description	Reset Value	Access
[31:2]	Reserved.	0x0000	RO

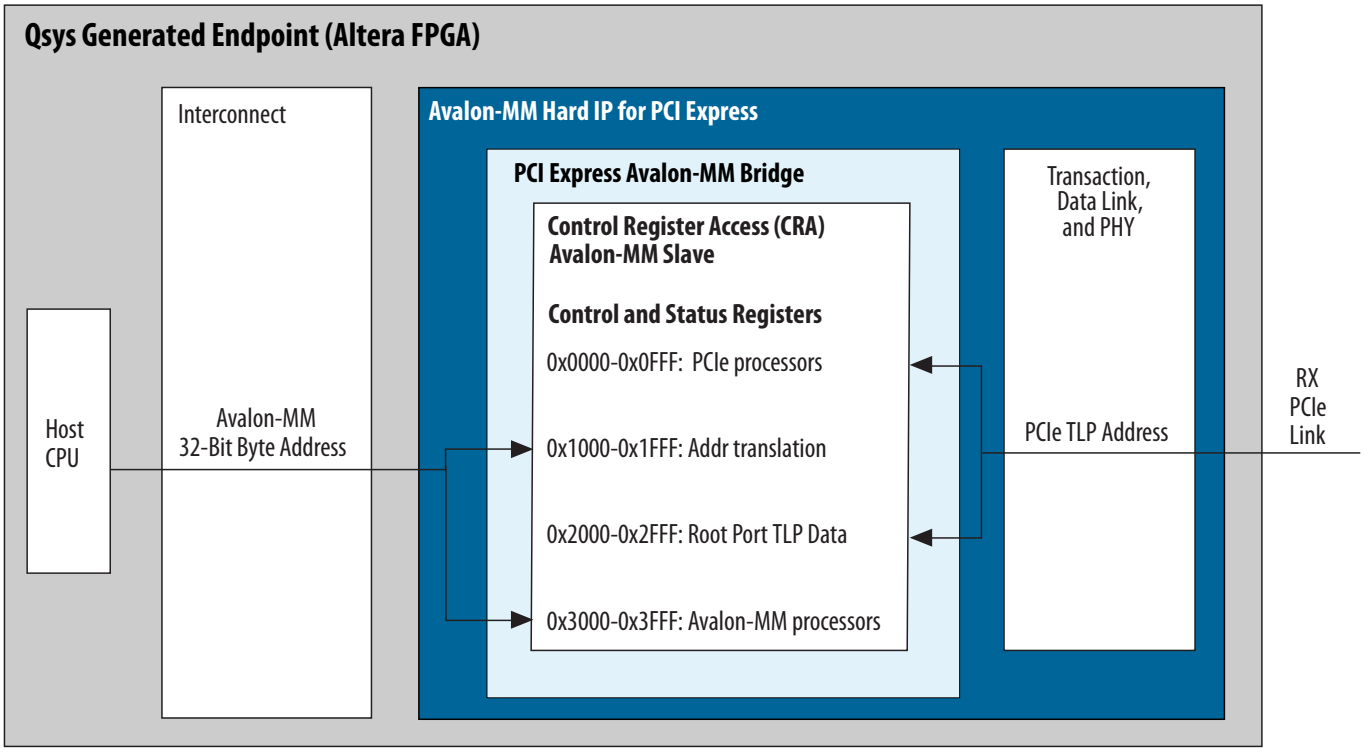
Bits	Register Description	Reset Value	Access
[1]	START_XFER. Sets the CvP output to the FPGA control block indicating the start of a transfer.	1'b0	RW
[0]	CVP_CONFIG. When asserted, instructs that the FPGA control block begin a transfer via CvP.	1'b0	RW

64- or 128-Bit Avalon-MM Bridge Register Descriptions

The CRA Avalon-MM slave module provides access control and status registers in the PCI Express Avalon-MM bridge. In addition, it provides access to selected Configuration Space registers and link status registers in read-only mode. This module is optional. However, you must include it to access the registers.

The control and status register address space is 16 KBytes. Each 4-KByte sub-region contains a set of functions, which may be specific to accesses from the PCI Express Root Complex only, from Avalon-MM processors only, or from both types of processors. Because all accesses come across the interconnect fabric —requests from the Avalon-MM Arria V Hard IP for PCI Express are routed through the interconnect fabric—hardware does not enforce restrictions to limit individual processor access to specific regions. However, the regions are designed to enable straight-forward enforcement by processor software. The following figure illustrates accesses to the Avalon-MM control and status registers from the Host CPU and PCI Express link.

Figure 5-9: Accesses to the Avalon-MM Bridge Control and Status Register



The following table describes the four subregions.

Table 5-11: Avalon-MM Control and Status Register Address Spaces

AddressRange	Address Space Usage
0x0000-0x0FFF	Registers typically intended for access by PCI Express link partner only. This includes PCI Express interrupt enable controls, write access to the PCI Express Avalon-MM bridge mailbox registers, and read access to Avalon-MM-to-PCI Express mailbox registers.
0x1000-0x1FFF	Avalon-MM-to-PCI Express address translation tables. Depending on the system design these may be accessed by the PCI Express link partner, Avalon-MM processors, or both.
0x2000-0x2FFF	Root Port request registers. An embedded processor, such as the Nios II processor, programs these registers to send the data for Configuration TLPs, I/O TLPs, single dword Memory Read and Write requests, and receive interrupts from an Endpoint.
0x3000-0x3FFF	Registers typically intended for access by Avalon-MM processors only. Provides host access to selected Configuration Space and status registers.

Note: The data returned for a read issued to any undefined address in this range is unpredictable.

The following table lists the complete address map for the PCI Express Avalon-MM bridge registers.

Note: In the following table the text in green are links to the detailed register description

Table 5-12: PCI Express Avalon-MM Bridge Register Map

Address Range	Register
0x0040	Avalon-MM to PCI Express Interrupt Status Register
0x0050	Avalon-MM to PCI Express Interrupt Status Enable Register
0x0800–0x081F	PCI Express-to-Avalon-MM Mailbox Registers
0x0900–x091F	Avalon-MM to PCI Express Mailbox Registers
0x1000–0x1FFF	Avalon-MM to PCI Express Address Translation Table
0x2000–0x2FFF	Root Port TLP Data Registers
0x3060	Avalon-MM to PCI Express Interrupt Status Registers for Root Ports
0x3060	PCI Express to Avalon-MM Interrupt Status Register for Endpoints

Address Range	Register
0x3070	INT-X Interrupt Enable Register for Root Ports
0x3070	INT-X Interrupt Enable Register for Endpoints
0x3A00-0x3A1F	Avalon-MM to PCI Express Mailbox Registers
0x3B00-0x3B1F	PCI Express to Avalon-MM Mailbox Registers
0x3C00-0x3C6C	Host (Avalon-MM master) access to selected Configuration Space and status registers.

Avalon-MM to PCI Express Interrupt Registers

Avalon-MM to PCI Express Interrupt Status Registers

These registers contain the status of various signals in the PCI Express Avalon-MM bridge logic and allow PCI Express interrupts to be asserted when enabled. Only Root Complexes should access these registers; however, hardware does not prevent other Avalon-MM masters from accessing them.

Table 5-13: Avalon-MM to PCI Express Interrupt Status Register, 0x0040

Bit	Name	Access	Description
[31:24]	Reserved	N/A	N/A
[23]	A2P_MAILBOX_INT7	RW1C	1 when the A2P_MAILBOX7 is written to
[22]	A2P_MAILBOX_INT6	RW1C	1 when the A2P_MAILBOX6 is written to
[21]	A2P_MAILBOX_INT5	RW1C	1 when the A2P_MAILBOX5 is written to
[20]	A2P_MAILBOX_INT4	RW1C	1 when the A2P_MAILBOX4 is written to
[19]	A2P_MAILBOX_INT3	RW1C	1 when the A2P_MAILBOX3 is written to
[18]	A2P_MAILBOX_INT2	RW1C	1 when the A2P_MAILBOX2 is written to
[17]	A2P_MAILBOX_INT1	RW1C	1 when the A2P_MAILBOX1 is written to
[16]	A2P_MAILBOX_INT0	RW1C	1 when the A2P_MAILBOX0 is written to

Bit	Name	Access	Description
[15:0]	AVL_IRQ_ASSERTED[15:0]	RO	Current value of the Avalon-MM interrupt (IRQ) input ports to the Avalon-MM RX master port: • 0—Avalon-MM IRQ is not being signaled. • 1—Avalon-MM IRQ is being signaled. A Qsys-generated IP Compiler for PCI Express has as many as 16 distinct IRQ input ports. Each AVL_IRQ_ASSERTED[] bit reflects the value on the corresponding IRQ input port.

Avalon-MM to PCI Express Interrupt Enable Registers

A PCI Express interrupt can be asserted for any of the conditions registered in the Avalon-MM to PCI Express Interrupt Status register by setting the corresponding bits in the Avalon-MM-to-PCI Express Interrupt Enable register. Either MSI or legacy interrupts can be generated as explained in the section *Enabling MSI or Legacy Interrupts*

Table 5-14: Avalon-MM to PCI Express Interrupt Enable Register, 0x0050

Bits	Name	Access	Description
[31:24]	Reserved	N/A	N/A
[23:16]	A2P_MB_IRQ	RW	Enables generation of PCI Express interrupts when a specified mailbox is written to by an external Avalon-MM master.
[4:0]	AVL_IRQ[15:0]	RW	Enables generation of PCI Express interrupts when a specified Avalon-MM interrupt signal is asserted. Your Qsys system may have as many as 16 individual input interrupt signals.

Table 5-15: Avalon-MM Interrupt Vector Register - 0x0060

Bits	Name	Access	Description
[31:5]	Reserved	N/A	N/A

Bits	Name	Access	Description
[15:0]	AVL_IRQ_Vector	RO	Stores the interrupt vector of the system interconnect fabric. The host software should read this register after being interrupted and determine the servicing priority.

PCI Express Mailbox Registers

The PCI Express Root Complex typically requires write access to a set of PCI Express-to-Avalon-MM mailbox registers and read-only access to a set of Avalon-MM-to-PCI Express mailbox registers. Eight mailbox registers are available.

The PCI Express-to-Avalon-MM Mailbox registers are writable at the addresses shown in the following table. Writing to one of these registers causes the corresponding bit in the Avalon-MM Interrupt Status register to be set to a one.

Table 5-16: PCI Express-to-Avalon-MM Mailbox Registers, 0x0800–0x081F

Address	Name	Access	Description
0x0800	P2A_MAILBOX0	RW	PCI Express-to-Avalon-MM Mailbox 0
0x0804	P2A_MAILBOX1	RW	PCI Express-to-Avalon-MM Mailbox 1
0x0808	P2A_MAILBOX2	RW	PCI Express-to-Avalon-MM Mailbox 2
0x080C	P2A_MAILBOX3	RW	PCI Express-to-Avalon-MM Mailbox 3
0x0810	P2A_MAILBOX4	RW	PCI Express-to-Avalon-MM Mailbox 4
0x0814	P2A_MAILBOX5	RW	PCI Express-to-Avalon-MM Mailbox 5
0x0818	P2A_MAILBOX6	RW	PCI Express-to-Avalon-MM Mailbox 6
0x081C	P2A_MAILBOX7	RW	PCI Express-to-Avalon-MM Mailbox 7

The Avalon-MM-to-PCI Express Mailbox registers are read at the addresses shown in the following table. The PCI Express Root Complex should use these addresses to read the mailbox information after being signaled by the corresponding bits in the Avalon MM to PCI Express Interrupt Status register.

Table 5-17: Avalon-MM-to-PCI Express Mailbox Registers, 0x0900–0x091F

Address	Name	Access	Description
0x0900	A2P_MAILBOX0	RO	Avalon-MM-to-PCI Express Mailbox 0

Address	Name	Access	Description
0x0904	A2P_MAILBOX1	RO	Avalon-MM-to-PCI Express Mailbox 1
0x0908	A2P_MAILBOX2	RO	Avalon-MM-to-PCI Express Mailbox 2
0x090C	A2P_MAILBOX3	RO	Avalon-MM-to-PCI Express Mailbox 3
0x0910	A2P_MAILBOX4	RO	Avalon-MM-to-PCI Express Mailbox 4
0x0914	A2P_MAILBOX5	RO	Avalon-MM-to-PCI Express Mailbox 5
0x0918	A2P_MAILBOX6	RO	Avalon-MM-to-PCI Express Mailbox 6
0x091C	A2P_MAILBOX7	RO	Avalon-MM-to-PCI Express Mailbox 7

Avalon-MM-to-PCI Express Address Translation Table

The Avalon-MM-to-PCI Express address translation table is writable using the CRA slave port. Each entry in the PCI Express address translation table is 8 bytes wide, regardless of the value in the current PCI Express address width parameter. Therefore, register addresses are always the same width, regardless of PCI Express address width.

These table entries are repeated for each address specified in the **Number of address pages** parameter. If **Number of address pages** is set to the maximum of 512, 0x1FF8 contains A2P_ADDR_MAP_LO511 and 0x1FFC contains A2P_ADDR_MAP_HI511.

Table 5-18: Avalon-MM-to-PCI Express Address Translation Table, 0x1000–0x1FFF

Address	Bits	Name	Access	Description
0x1000	[1:0]	A2P_ADDR_SPACE0	RW	Address space indication for entry 0. Refer to Table 9–31 for the definition of these bits.
	[31:2]	A2P_ADDR_MAP_LO0	RW	Lower bits of Avalon-MM-to-PCI Express address map entry 0.
0x1004	[31:0]	A2P_ADDR_MAP_HI0	RW	Upper bits of Avalon-MM-to-PCI Express address map entry 0.

Address	Bits	Name	Access	Description
0x1008	[1:0]	A2P_ADDR_SPACE1	RW	Address space indication for entry 1. Refer to the following encodings are defined: • 2'b00:. Memory Space, 32-bit PCI Express address. 32-bit header is generated. Address bits 63:32 of the translation table entries are ignored. • 2'b01: Memory space, 64-bit PCI Express address. 64-bit address header is generated. • 2'b10: Reserved • 2'b11: Reserved
	[31:2]	A2P_ADDR_MAP_LO1	RW	Lower bits of Avalon-MM-to-PCI Express address map entry 1. This entry is only implemented if the number of address translation table entries is greater than 1.
0x100C	[31:0]	A2P_ADDR_MAP_HI1	RW	Upper bits of Avalon-MM-to-PCI Express address map entry 1. This entry is only implemented if the number of address translation table entries is greater than 1.

PCI Express to Avalon-MM Interrupt Status and Enable Registers for Endpoints

The registers in this section contain status of various signals in the PCI Express Avalon-MM bridge logic and allow Avalon interrupts to be asserted when enabled. A processor local to the interconnect fabric that processes the Avalon-MM interrupts can access these registers.

Note: These registers must not be accessed by the PCI Express Avalon-MM bridge master ports; however, there is nothing in the hardware that prevents a PCI Express Avalon-MM bridge master port from accessing these registers.

The following table describes the Interrupt Status register when you configure the core as an Endpoint. It records the status of all conditions that can cause an Avalon-MM interrupt to be asserted.

Table 5-19: PCI Express to Avalon-MM Interrupt Status Register for Endpoints, 0x3060

Bits	Name	Access	Description
0	ERR_PCI_WRITE_FAILURE	RW1C	When set to 1, indicates a PCI Express write failure. This bit can also be cleared by writing a 1 to the same bit in the Avalon MM to PCI Express Interrupt Status register.

Bits	Name	Access	Description
1	ERR_PCI_READ_FAILURE	RW1C	When set to 1, indicates the failure of a PCI Express read. This bit can also be cleared by writing a 1 to the same bit in the Avalon MM to PCI Express Interrupt Status register.
[15:2]	Reserved	—	—
[16]	P2A_MAILBOX_INT0	RW1C	1 when the P2A_MAILBOX0 is written
[17]	P2A_MAILBOX_INT1	RW1C	1 when the P2A_MAILBOX1 is written
[18]	P2A_MAILBOX_INT2	RW1C	1 when the P2A_MAILBOX2 is written
[19]	P2A_MAILBOX_INT3	RW1C	1 when the P2A_MAILBOX3 is written
[20]	P2A_MAILBOX_INT4	RW1C	1 when the P2A_MAILBOX4 is written
[21]	P2A_MAILBOX_INT5	RW1C	1 when the P2A_MAILBOX5 is written
[22]	P2A_MAILBOX_INT6	RW1C	1 when the P2A_MAILBOX6 is written
[23]	P2A_MAILBOX_INT7	RW1C	1 when the P2A_MAILBOX7 is written
[31:24]	Reserved	—	—

An Avalon-MM interrupt can be asserted for any of the conditions noted in the Avalon-MM Interrupt Status register by setting the corresponding bits in the PCI Express to Avalon-MM Interrupt Enable register.

PCI Express interrupts can also be enabled for all of the error conditions described. However, it is likely that only one of the Avalon-MM or PCI Express interrupts can be enabled for any given bit. Typically, a single process in either the PCI Express or Avalon-MM domain handles the condition reported by the interrupt.

Table 5-20: INT-X Interrupt Enable Register for Endpoints, 0x3070

Bits	Name	Access	Description
[31:0]	PCI Express to Avalon-MM Interrupt Enable	RW	When set to 1, enables the interrupt for the corresponding bit in the PCI Express to Avalon MM Interrupt Status register to cause the Avalon Interrupt signal (cra_Irq_o) to be asserted. Only bits implemented in the PCI Express to Avalon MM Interrupt Status register are implemented in the Enable register. Reserved bits cannot be set to a 1.

Avalon-MM Mailbox Registers

A processor local to the interconnect fabric typically requires write access to a set of Avalon-MM-to-PCI Express Mailbox registers and read-only access to a set of PCI Express-to-Avalon-MM Mailbox registers. Eight mailbox registers are available.

The Avalon-MM-to-PCI Express Mailbox registers are writable at the addresses shown in the following table. When the Avalon-MM processor writes to one of these registers the corresponding bit in the Avalon MM to PCI Express Interrupt Status register is set to 1.

Table 5-21: Avalon-MM to PCI Express Mailbox Registers, 0x3A00–0x3A1F

Address	Name	Access	Description
0x3A00	A2P_MAILBOX0	RW	Avalon-MM-to-PCI Express mailbox 0
0x3A04	A2P_MAILBOX1	RW	Avalon-MM-to-PCI Express mailbox 1
0x3A08	A2P _MAILBOX2	RW	Avalon-MM-to-PCI Express mailbox 2
0x3A0C	A2P _MAILBOX3	RW	Avalon-MM-to-PCI Express mailbox 3
0x3A10	A2P _MAILBOX4	RW	Avalon-MM-to-PCI Express mailbox 4
0x3A14	A2P _MAILBOX5	RW	Avalon-MM-to-PCI Express mailbox 5
0x3A18	A2P _MAILBOX6	RW	Avalon-MM-to-PCI Express mailbox 6
0x3A1C	A2P_MAILBOX7	RW	Avalon-MM-to-PCI Express mailbox 7

The PCI Express-to-Avalon-MM Mailbox registers are read-only at the addresses shown in the following table. The Avalon-MM processor reads these registers when the corresponding bit in the PCI Express to Avalon-MM Interrupt Status register is set to 1.

Table 5-22: PCI Express to Avalon-MM Mailbox Registers, 0x3B00–0x3B1F

Address	Name	Access Mode	Description
0x3B00	P2A_MAILBOX0	RO	PCI Express-to-Avalon-MM mailbox 0
0x3B04	P2A_MAILBOX1	RO	PCI Express-to-Avalon-MM mailbox 1
0x3B08	P2A_MAILBOX2	RO	PCI Express-to-Avalon-MM mailbox 2
0x3B0C	P2A_MAILBOX3	RO	PCI Express-to-Avalon-MM mailbox 3
0x3B10	P2A_MAILBOX4	RO	PCI Express-to-Avalon-MM mailbox 4
0x3B14	P2A_MAILBOX5	RO	PCI Express-to-Avalon-MM mailbox 5
0x3B18	P2A_MAILBOX6	RO	PCI Express-to-Avalon-MM mailbox 6
0x3B1C	P2A_MAILBOX7	RO	PCI Express-to-Avalon-MM mailbox 7

Control Register Access (CRA) Avalon-MM Slave Port

Table 5-23: Configuration Space Register Descriptions

For registers that are less than 32 bits, the upper bits are unused.

Byte Offset	Register	Dir	Description
14'h3C00	cfg_dev_ctrl[15:0]	O	cfg_devctrl[15:0] is device control for the PCI Express capability structure.
14'h3C04	cfg_dev_ctrl2[15:0]	O	cfg_dev2ctrl[15:0] is device control 2 for the PCI Express capability structure.

Byte Offset	Register	Dir	Description
14'h3C08	cfg_link_ctrl[15:0]	O	cfg_link_ctrl[15:0]is the primary Link Control of the PCI Express capability structure. For Gen2 or Gen3 operation, you must write a 1'b1 to Retrain Link bit (Bit[5] of the cfg_link_ctrl) of the Root Port to initiate retraining to a higher data rate after the initial link training to Gen1 L0 state. Retraining directs the LTSSM to the Recovery state. Retraining to a higher data rate is not automatic for the Arria V Hard IP for PCI Express IP Core even if both devices on the link are capable of a higher data rate.
14'h3C0C	cfg_link_ctrl2[15:0]	O	cfg_link_ctrl2[31:16] is the secondary Link Control register of the PCI Express capability structure for Gen2 operation. When tl_cfg_addr=2, tl_cfg_ctl returns the primary and secondary Link Control registers, {cfg_link_ctrl[15:0], cfg_link_ctrl2[15:0]}, the primary Link Status register contents is available on tl_cfg_sts[46:31]. For Gen1 variants, the link bandwidth notification bit is always set to 0. For Gen2 variants, this bit is set to 1.
14'h3C10	cfg_prm_cmd[15:0]	O	Base/Primary Command register for the PCI Configuration Space.
14'h3C14	cfg_root_ctrl[7:0]	O	Root control and status register of the PCI-Express capability. This register is only available in Root Port mode.
14'h3C18	cfg_sec_ctrl[15:0]	O	Secondary bus Control and Status register of the PCI-Express capability. This register is only available in Root Port mode.
14'h3C1C	cfg_secbus[7:0]	O	Secondary bus number. Available in Root Port mode.
14'h3C20	cfg_subbus[7:0]	O	Subordinate bus number. Available in Root Port mode.
14'h3C24	cfg_msi_addr_low[31:0]	O	cfg_msi_add[31:0] is the MSI message address.

Byte Offset	Register	Dir	Description
14'h3C28	cfg_msi_addr_hi[63:32]	O	cfg_msi_add[63:32] is the MSI upper message address.
14'h3C2C	cfg_io_bas[19:0]	O	The IO base register of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C30	cfg_io_lim[19:0]	O	The IO limit register of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C34	cfg_np_bas[11:0]	O	The non-prefetchable memory base register of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C38	cfg_np_lim[11:0]	O	The non-prefetchable memory limit register of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C3C	cfg_pr_bas_low[31:0]	O	The lower 32 bits of the prefetchable base register of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C40	cfg_pr_bas_hi[43:32]	O	The upper 12 bits of the prefetchable base registers of the Type1 Configuration Space. This register is only available in Root Port mode.
14'h3C44	cfg_pr_lim_low[31:0]	O	The lower 32 bits of the prefetchable limit registers of the Type1 Configuration Space. Available in Root Port mode.
14'h3C48	cfg_pr_lim_hi[43:32]	O	The upper 12 bits of the prefetchable limit registers of the Type1 Configuration Space. Available in Root Port mode.
14'h3C4C	cfg_pmcsr[31:0]	O	cfg_pmcsr[31:16] is Power Management Control and cfg_pmcsr[15:0]is the Power Management Status register.
14'h3C50	cfg_msixcsr[15:0]	O	MSI-X message control register.
14'h3C54	cfg_msicsr[15:0]	O	MSI message control.

Byte Offset	Register	Dir	Description
14'h3C58	cfg_tcvcmap[23:0]	O	Configuration traffic class (TC)/virtual channel (VC) mapping. The Application Layer uses this signal to generate a TLP mapped to the appropriate channel based on the traffic class of the packet. The following encodings are defined: • cfg_tcvcmap[2:0]: Mapping for TC0 (always 0) . • cfg_tcvcmap[5:3]: Mapping for TC1. • cfg_tcvcmap[8:6]: Mapping for TC2. • cfg_tcvcmap[11:9]: Mapping for TC3. • cfg_tcvcmap[14:12]: Mapping for TC4. • cfg_tcvcmap[17:15]: Mapping for TC5. • cfg_tcvcmap[20:18]: Mapping for TC6. • cfg_tcvcmap[23:21]: Mapping for TC7.
14'h3C5C	cfg_msi_data[15:0]	O	cfg_msi_data[15:0] is message data for MSI.
14'h3C60	cfg_busdev[12:0]	O	Bus/Device Number captured by or programmed in the Hard IP.
14'h3C64	ltssm_reg[4:0]	O	Specifies the current LTSSM state. The LTSSM state machine encoding defines the following states: • 00000: Detect.Quiet • 00001: Detect.Active • 00010: Polling.Active • 00011: Polling.Compliance • 00100: Polling.Configuration • 00101: Polling.Speed • 00110: config.Linkwidthstart • 00111: Config.Linkaccept • 01000: Config.Lanenumaccept • 01001: Config.Lanenumwait • 01010: Config.Complete • 01011: Config.Idle • 01100: Recovery.Rcvlock • 01101: Recovery.Rcvconfig • 01110: Recovery.Idle • 01111: L0 • 10000: Disable • 10001: Loopback.Entry • 10010: Loopback.Active

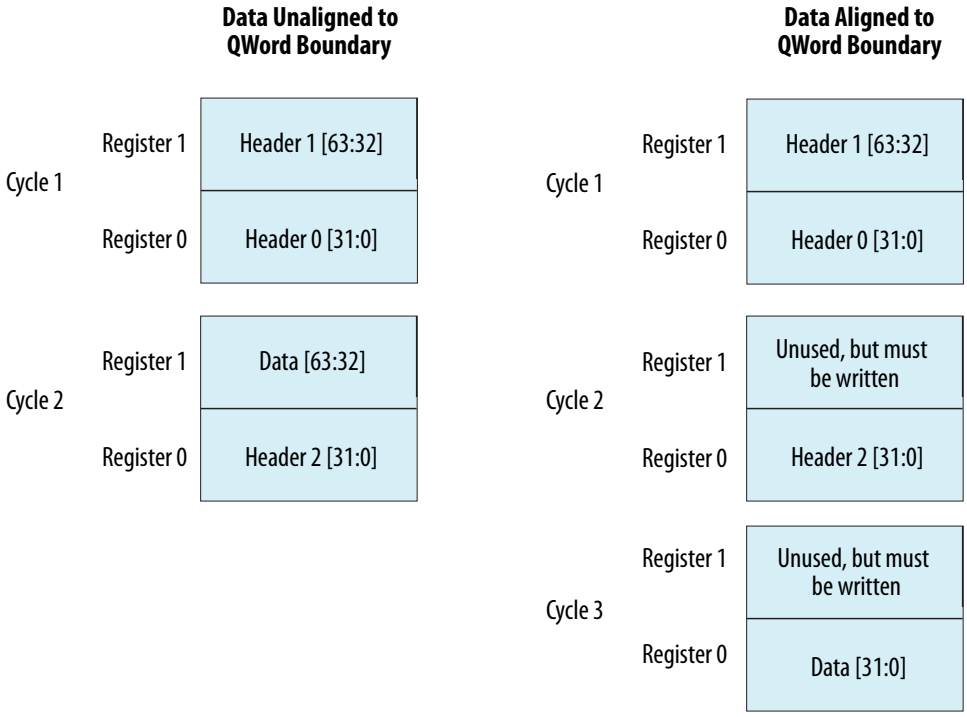
Byte Offset	Register	Dir	Description
			• 10011: Loopback.Exit • 10100: Hot.Reset • 10101: LOs • 11001: L2.transmit.Wake • 11010: Speed.Recovery • 11011: Recovery.Equalization, Phase 0 • 11100: Recovery.Equalization, Phase 1 • 11101: Recovery.Equalization, Phase 2 • 11110: recovery.Equalization, Phase 3
14'h3C68	`current_speed_reg[1:0]`	O	Indicates the current speed of the PCIe link. The following encodings are defined: • 2b'00: Undefined • 2b'01: Gen1 • 2b'10: Gen2 • 2b'11: Gen3
14'h3C6C	`lane_act_reg[3:0]`	O	Lane Active Mode: This signal indicates the number of lanes that configured during link training. The following encodings are defined: • 4'b0001: 1 lane • 4'b0010: 2 lanes • 4'b0100: 4 lanes • 4'b1000: 8 lanes

Programming Model for Avalon-MM Root Port

The Application Layer writes the Root Port TLP TX Data registers with TLP formatted data for Configuration Read and Write Requests, Message TLPs, I/O Read and Write Requests, or single dword Memory Read and Write Requests. Software should check the Root Port `Link Status` register (offset 0x92) to ensure the Data Link Layer Link `Active` bit is set to 1'b1 before issuing a Configuration request to downstream ports.

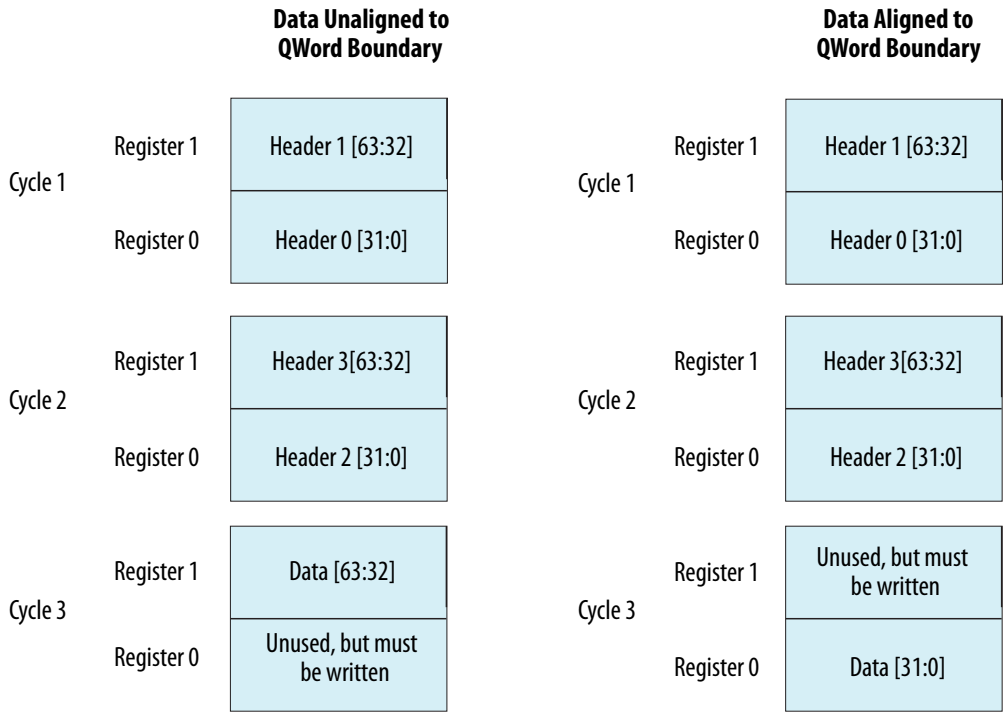
The Application Layer data must be in the appropriate TLP format with the data payload aligned to the TLP address. Aligning the payload data to the TLP address may result in the payload data being either aligned or unaligned to the qword. The following figure illustrates three dword TLPs with data that is aligned and unaligned to the qword.

Figure 5-10: Layout of Data with 3 Dword Headers



The following figure illustrates four dword TLPs with data that are aligned and unaligned to the qword.

Figure 5-11: Layout of Data with 4 Dword Headers



The TX TLP programming model scales with the data width. The Application Layer performs the same writes for both the 64- and 128-bit interfaces. The Application Layer can only have one outstanding non-posted request at a time. The Application Layer must use tags 16–31 to identify non-posted requests.

Note: For Root Ports, the Avalon-MM bridge does not filter Type 0 Configuration Requests by device number. Application Layer software should filter out all requests to Avalon-MM Root Port registers that are not for device 0. Application Layer software should return an Unsupported Request Completion Status.

Sending a Write TLP

The Application Layer performs the following sequence of Avalon-MM accesses to the CRA slave port to send a Memory Write Request:

1. Write the first 32 bits of the TX TLP to `RP_TX_REG0`.
2. Write the next 32 bits of the TX TLP to `RP_TX_REG1`.
3. Write the `RP_TX_CNTRL.SOP` to 1'b1 to push the first two dwords of the TLP into the Root Port TX FIFO.
4. Repeat Steps 1 and 2. The second write to `RP_TX_REG1` is required, even for three dword TLPs with aligned data.
5. If the packet is complete, write `RP_TX_CNTRL` to 2'b10 to indicate the end of the packet. If the packet is not complete, write 2'b00 to `RP_TX_CNTRL`.
6. Repeat this sequence to program a complete TLP.

When the programming of the TX TLP is complete, the Avalon-MM bridge schedules the TLP with higher priority than TX TLPs coming from the TX slave port.

Sending a Read TLP or Receiving a Non-Posted Completion TLP

The TLPs associated with the Non-Posted TX requests are stored in the RP_RX_CPL FIFO buffer and subsequently loaded into RP_RXCPL registers. The Application Layer performs the following sequence to retrieve the TLP.

1. Polls the `RP_RXCPL_STA TUS.SOP` to determine when it is set to 1'b1.
2. Then `RP_RXCPL_STATUS.SOP` = 1'b'1, reads `RP_RXCPL_REG0` and `RP_RXCPL_REG1` to retrieve dword 0 and dword 1 of the TLP.
3. Read the `RP_RXCPL_STATUS.EOP`.
   - If `RP_RXCPL_STATUS.EOP` = 1'b0, read `RP_RXCPL_REG0` and `RP_RXCPL_REG1` to retrieve dword 2 and dword 3 of the TLP, then repeat step 3.
   - If `RP_RXCPL_STATUS.EOP` = 1'b1, read `RP_RXCPL_REG0` and `RP_RXCPL_REG1` to retrieve final dwords of TLP.

PCI Express to Avalon-MM Interrupt Status and Enable Registers for Root Ports

The Root Port supports MSI, MSI-X and legacy (INTx) interrupts. MSI and MSI-X interrupts are memory writes from the Endpoint to the Root Port. MSI and MSI-X requests are forwarded to the interconnect without asserting `CraIrq_o`.

Table 5-24: Avalon-MM Interrupt Status Registers for Root Ports, 0x3060

Bits	Name	Access Mode	Description
[31:5]	Reserved	—	—
[4]	RPRX_CPL_RECEIVED	RW1C	Set to 1'b1 when the Root Port has received a Completion TLP for an outstanding Non-Posted request from the TLP Direct channel.
[3]	INTD_RECEIVED	RW1C	The Root Port has received INTD from the Endpoint.
[2]	INTC_RECEIVED	RW1C	The Root Port has received INTC from the Endpoint.
[1]	INTB_RECEIVED	RW1C	The Root Port has received INTB from the Endpoint.
[0]	INTA_RECEIVED	RW1C	The Root Port has received INTA from the Endpoint.

Table 5-25: INT-X Interrupt Enable Register for Root Ports, 0x3070

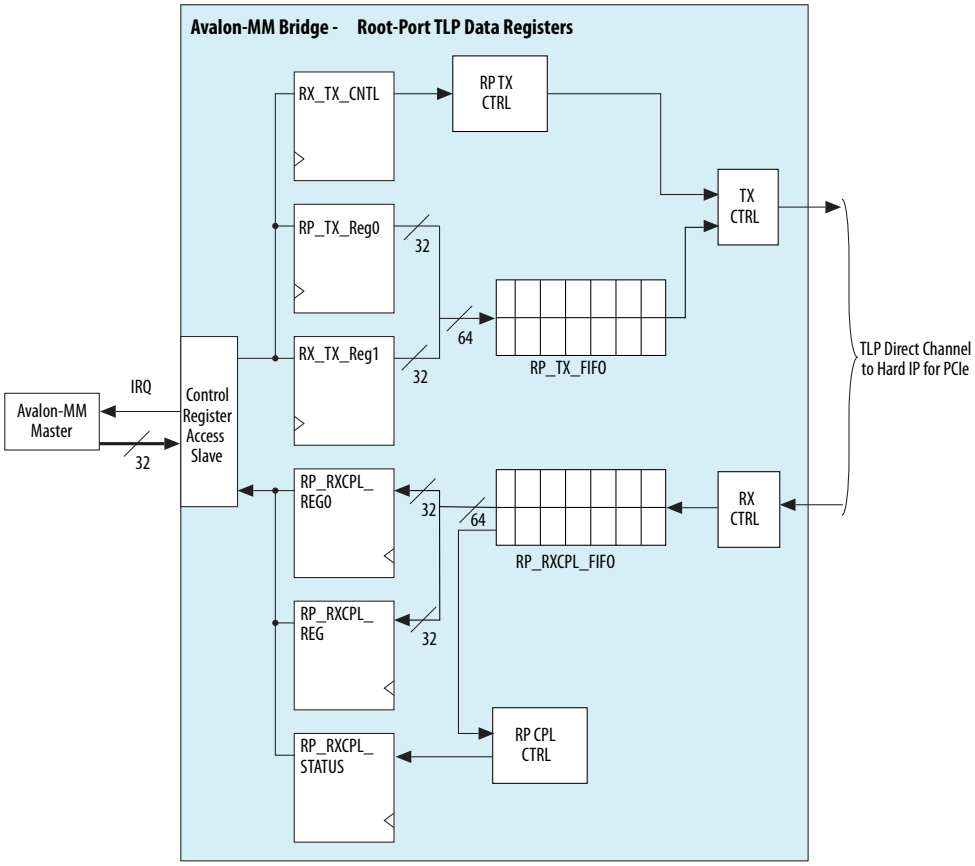
Bit	Name	Access Mode	Description
[31:5]	Reserved	—	—
[4]	RPRX_CPL_RECEIVED	RW	When set to 1'b1, enables the assertion of CraIrq_o when the Root Port Interrupt Status register RPRX_CPL_RECEIVED bit indicates it has received a Completion for a Non-Posted request from the TLP Direct channel.
[3]	INTD_RECEIVED_ENA	RW	When set to 1'b1, enables the assertion of CraIrq_o when the Root Port Interrupt Status register INTD_RECEIVED bit indicates it has received INTD.

Bit	Name	Access Mode	Description
[2]	INTC_RECEIVED_ENA	RW	When set to 1'b1, enables the assertion of CraIrq_o when the Root Port Interrupt Status register INTC_RECEIVED bit indicates it has received INTC.
[1]	INTB_RECEIVED_ENA	RW	When set to 1'b1, enables the assertion of CraIrq_o when the Root Port Interrupt Status register INTB_RECEIVED bit indicates it has received INTB.
[0]	INTA_RECEIVED_ENA	RW	When set to 1'b1, enables the assertion of CraIrq_o when the Root Port Interrupt Status register INTA_RECEIVED bit indicates it has received INTA.

Root Port TLP Data Registers

The TLP data registers provide a mechanism for the Application Layer to specify data that the Root Port uses to construct Configuration TLPs, I/O TLPs, and single dword Memory Reads and Write requests. The Root Port then drives the TLPs on the TLP Direct Channel to access the Configuration Space, I/O space, or Endpoint memory.

Figure 5-12: Root Port TLP Data Registers



Note: The high performance TLPs implemented by Avalon-MM ports in the Avalon-MM Bridge are also available for Root Ports. For more information about these TLPs, refer to *Avalon-MM Bridge TLPs*.

Table 5-26: Root Port TLP Data Registers, 0x2000–0x2FFF

Root-Port Request Registers				Address Range: 0x2800-0x2018
Address	Bits	Name	Access	Description
0x2000	[31:0]	RP_TX_REG0	W	Lower 32 bits of the TX TLP.
0x2004	[31:0]	RP_TX_REG1	W	Upper 32 bits of the TX TLP.
0x2008	[31:2]	Reserved	—	—
	[1]	RP_TX_CNTRL.EOP	W	Write 1'b1 to specify the of end a packet. Writing this bit frees the corresponding entry in the FIFO.
	[0]	RP_TX_CNTRL.SOP	W	Write 1'b1 to specify the start of a packet.

Root-Port Request Registers				Address Range: 0x2800-0x2018
Address	Bits	Name	Access	Description
0x2010	[31:16]	Reserved	—	—
	[15:8]	RP_RXCPL_STATUS	R	Specifies the number of words in the RX completion FIFO that contain valid data.
	[7:2]	Reserved	—	—
	[1]	RP_RXCPL_STATUS.EOP	R	When 1'b1, indicates that the data for a Completion TLP is ready to be read by the Application Layer. The Application Layer must poll this bit to determine when a Completion TLP is available.
	[0]	RP_RXCPL_STATUS.SOP	R	When 1'b1, indicates that the final data for a Completion TLP is ready to be read by the Application Layer. The Application Layer must poll this bit to determine when the final data for a Completion TLP is available.
0x2014	[31:0]	RP_RXCPL_REG1	RC	Lower 32 bits of a Completion TLP. Reading frees this entry in the FIFO.
0x2018	[31:0]	RP_RXCPL_REG1	RC	Upper 32 bits of a Completion TLP. Reading frees this entry in the FIFO.

Related Information

Avalon-MM Bridge TLPs on page 9-11

Uncorrectable Internal Error Mask Register

Table 5-27: Uncorrectable Internal Error Mask Register

The `Uncorrectable Internal Error Mask` register controls which errors are forwarded as internal uncorrectable errors. With the exception of the configuration error detected in CvP mode, all of the errors are severe and may place the device or PCIe link in an inconsistent state. The configuration error detected in CvP mode may be correctable depending on the design of the programming software. The access code *RWS* stands for Read Write Sticky meaning the value is retained after a soft reset of the IP core.

Bits	Register Description	Reset Value	Access
[31:12]	Reserved.	1b'0	RO

Bits	Register Description	Reset Value	Access
[11]	Mask for RX buffer posted and completion overflow error.	1b'1	RWS
[10]	Reserved	1b'0	RO
[9]	Mask for parity error detected on Configuration Space to TX bus interface.	1b'1	RWS
[8]	Mask for parity error detected on the TX to Configuration Space bus interface.	1b'1	RWS
[7]	Mask for parity error detected at TX Transaction Layer error.	1b'1	RWS
[6]	Reserved	1b'0	RO
[5]	Mask for configuration errors detected in CvP mode.	1b'0	RWS
[4]	Mask for data parity errors detected during TX Data Link LCRC generation.	1b'1	RWS
[3]	Mask for data parity errors detected on the RX to Configuration Space Bus interface.	1b'1	RWS
[2]	Mask for data parity error detected at the input to the RX Buffer.	1b'1	RWS
[1]	Mask for the retry buffer uncorrectable ECC error.	1b'1	RWS
[0]	Mask for the RX buffer uncorrectable ECC error.	1b'1	RWS

Uncorrectable Internal Error Status Register

Table 5-28: Uncorrectable Internal Error Status Register

This register reports the status of the internally checked errors that are uncorrectable. When specific errors are enabled by the `Uncorrectable Internal Error Mask` register, they are handled as Uncorrectable Internal Errors as defined in the *PCI Express Base Specification 3.0*. This register is for debug only. It should only be used to observe behavior, not to drive custom logic. The access code RW1CS represents Read Write 1 to Clear Sticky.

Bits	Register Description	Reset Value	Access
[31:12]	Reserved.	0	RO
[11]	When set, indicates an RX buffer overflow condition in a posted request or Completion	0	RW1CS

Bits	Register Description	Reset Value	Access
[10]	Reserved.	0	RO
[9]	When set, indicates a parity error was detected on the Configuration Space to TX bus interface	0	RW1CS
[8]	When set, indicates a parity error was detected on the TX to Configuration Space bus interface	0	RW1CS
[7]	When set, indicates a parity error was detected in a TX TLP and the TLP is not sent.	0	RW1CS
[6]	When set, indicates that the Application Layer has detected an uncorrectable internal error.	0	RW1CS
[5]	When set, indicates a configuration error has been detected in CvP mode which is reported as uncorrectable. This bit is set whenever a `CVP_CONFIG_ERROR` rises while in `CVP_MODE`.	0	RW1CS
[4]	When set, indicates a parity error was detected by the TX Data Link Layer.	0	RW1CS
[3]	When set, indicates a parity error has been detected on the RX to Configuration Space bus interface.	0	RW1CS
[2]	When set, indicates a parity error was detected at input to the RX Buffer.	0	RW1CS
[1]	When set, indicates a retry buffer uncorrectable ECC error.	0	RW1CS
[0]	When set, indicates a RX buffer uncorrectable ECC error.	0	RW1CS

Correctable Internal Error Mask Register

Table 5-29: Correctable Internal Error Mask Register

The `Correctable Internal Error Mask` register controls which errors are forwarded as Internal Correctable Errors. This register is for debug only.

Bits	Register Description	Reset Value	Access
[31:7]	Reserved.	0	RO
[6]	Mask for Corrected Internal Error reported by the Application Layer.	1	RWS

Bits	Register Description	Reset Value	Access
[5]	Mask for configuration error detected in CvP mode.	0	RWS
[4:2]	Reserved.	0	RO
[1]	Mask for retry buffer correctable ECC error.	1	RWS
[0]	Mask for RX Buffer correctable ECC error.	1	RWS

Correctable Internal Error Status Register

Table 5-30: Correctable Internal Error Status Register

The `Correctable Internal Error Status` register reports the status of the internally checked errors that are correctable. When these specific errors are enabled by the `Correctable Internal Error Mask` register, they are forwarded as Correctable Internal Errors as defined in the *PCI Express Base Specification 3.0*. This register is for debug only. It should only be used to observe behavior, not to drive logic custom logic.

Bits	Register Description	Reset Value	Access
[31:6]	Reserved.	0	RO
[5]	When set, indicates a configuration error has been detected in CvP mode which is reported as correctable. This bit is set whenever a `CVP_CONFIG_ERROR` occurs while in `CVP_MODE`.	0	RW1CS
[4:2]	Reserved.	0	RO
[1]	When set, the retry buffer correctable ECC error status indicates an error.	0	RW1CS
[0]	When set, the RX buffer correctable ECC error status indicates an error.	0	RW1CS

Reset and Clocks

6

2014.12.15

UG-01105_avmm

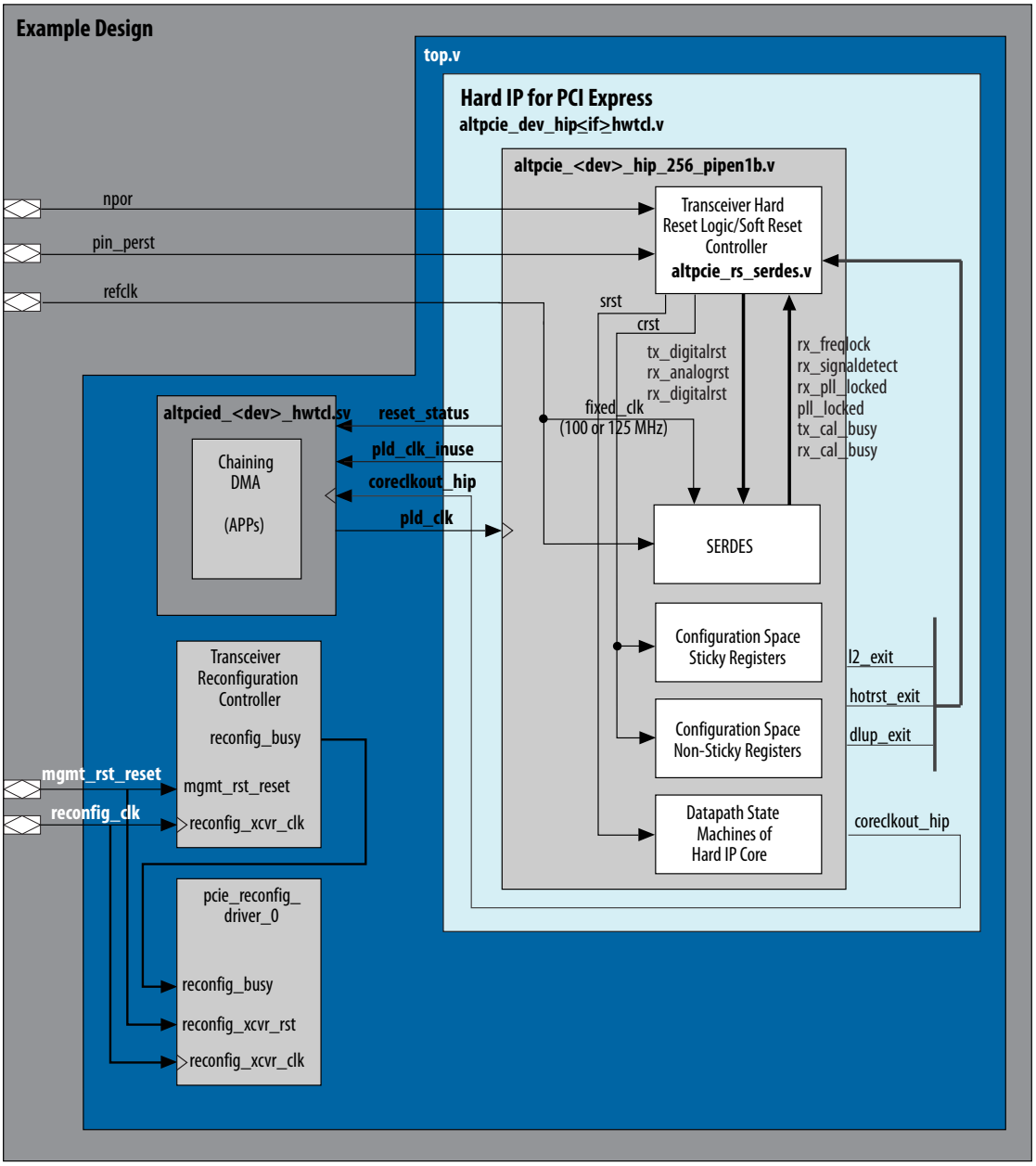
Subscribe Send Feedback

The `pin_perst` signal from the input pin of the FPGA resets the Hard IP for PCI Express IP Core. `app_rstn` which resets the Application Layer logic is derived from `reset_status` and `pld_clk_inuse`, which are outputs of the core. This reset controller is implemented in hardened logic. The figure below provides a simplified view of the logic that implements the reset controller.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

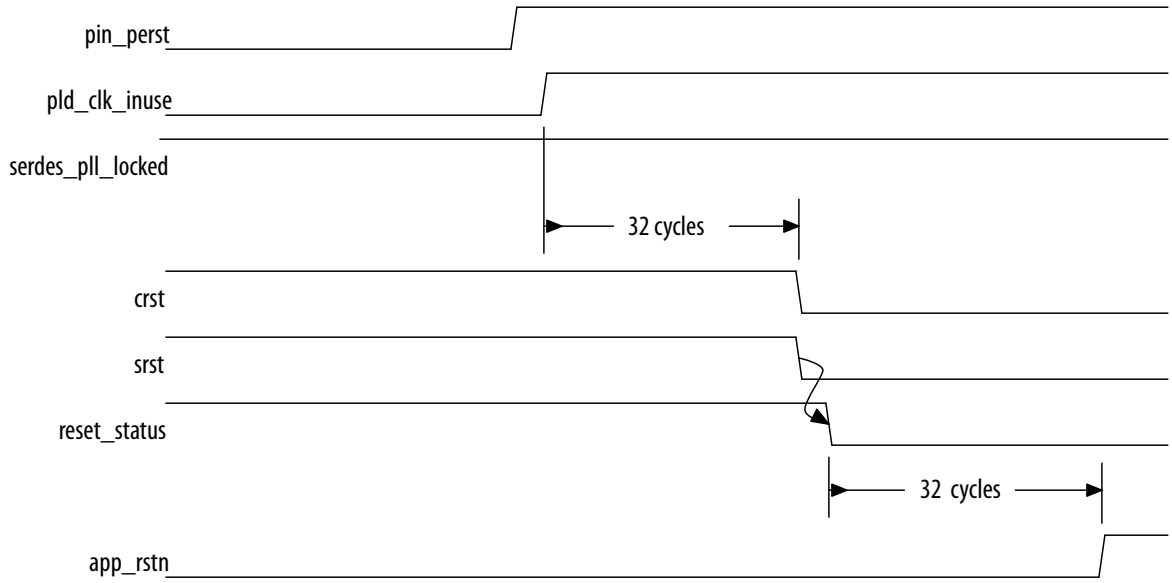
Figure 6-1: Reset Controller Block Diagram



Reset Sequence for Hard IP for PCI Express IP Core and Application Layer

Figure 6-2: Hard IP for PCI Express and Application Logic Reset Sequence

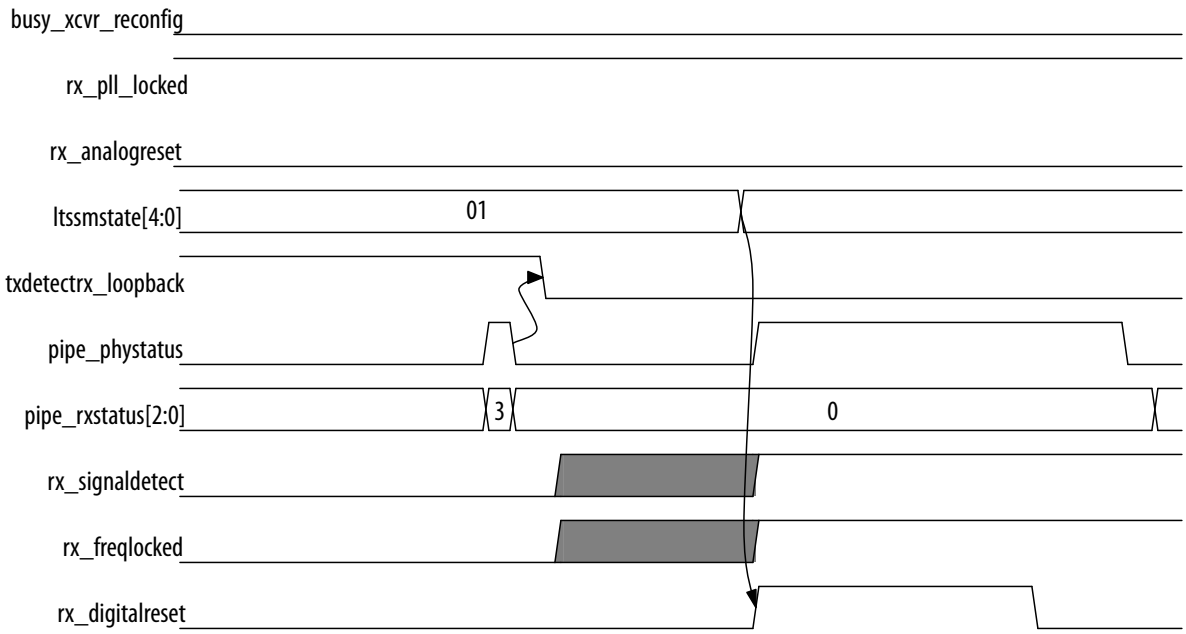
Your Application Layer can instantiate a module similar to the one in this figure to generate `app_rstn`, which resets the Application Layer logic.



This reset sequence includes the following steps:

1. After `pin_perst` or `npor` is released, the Hard IP reset controller waits for `pld_clk_inuse` to be asserted.
2. `csrt` and `srst` are released 32 cycles after `pld_clk_inuse` is asserted.
3. The Hard IP for PCI Express deasserts the `reset_status` output to the Application Layer.
4. The **altpcied_<*device*>v_hwtcl.sv** deasserts `app_rstn` 32 `pld_clk`cycles after `reset_status` is released.

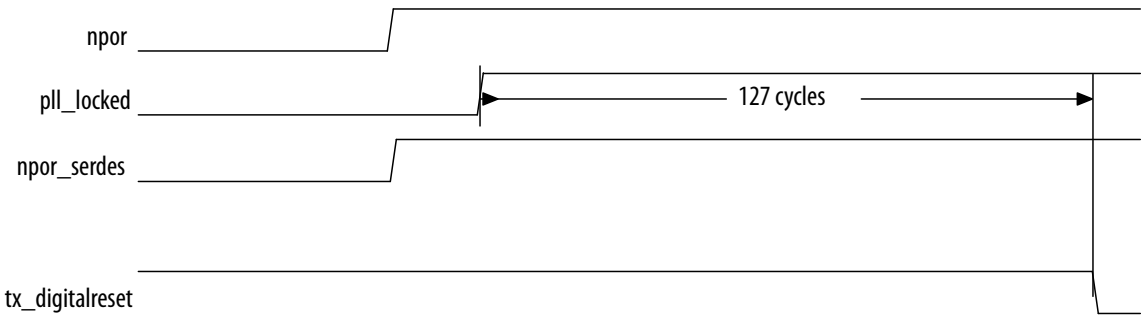
Figure 6-3: RX Transceiver Reset Sequence



The RX transceiver reset sequence includes the following steps:

1. After `rx_pll_locked` is asserted, the LTSSM state machine transitions from the Detect.Quiet to the Detect.Active state.
2. When the `pipe_phystatus` pulse is asserted and `pipe_rxstatus[2:0]` = 3, the receiver detect operation has completed.
3. The LTSSM state machine transitions from the Detect.Active state to the Polling.Active state.
4. The Hard IP for PCI Express asserts `rx_digitalreset`. The `rx_digitalreset` signal is deasserted after `rx_signaldetect` is stable for a minimum of 3 ms.

Figure 6-4: TX Transceiver Reset Sequence



The TX transceiver reset sequence includes the following steps:

1. After `npor` is deasserted, the IP core deasserts the `npor_serdes` input to the TX transceiver.
2. The SERDES reset controller waits for `pll_locked` to be stable for a minimum of 127 `pld_clk` cycles before deasserting `tx_digitalreset`.

For descriptions of the available reset signals, refer to *Reset Signals, Status, and Link Training Signals*.

Clocks

The Hard IP contains a clock domain crossing (CDC) synchronizer at the interface between the PHY/MAC and the DLL layers. The synchronizer allows the Data Link and Transaction Layers to run at frequencies independent of the PHY/MAC. The CDC synchronizer provides more flexibility for the user clock interface. Depending on parameters you specify, the core selects the appropriate `coreclkout_hip`. You can use these parameters to enhance performance by running at a higher frequency for latency optimization or at a lower frequency to save power.

In accordance with the *PCI Express Base Specification*, you must provide a 100 MHz reference clock that is connected directly to the transceiver.

As a convenience, you may also use a 125 MHz input reference clock as input to the TX PLL.

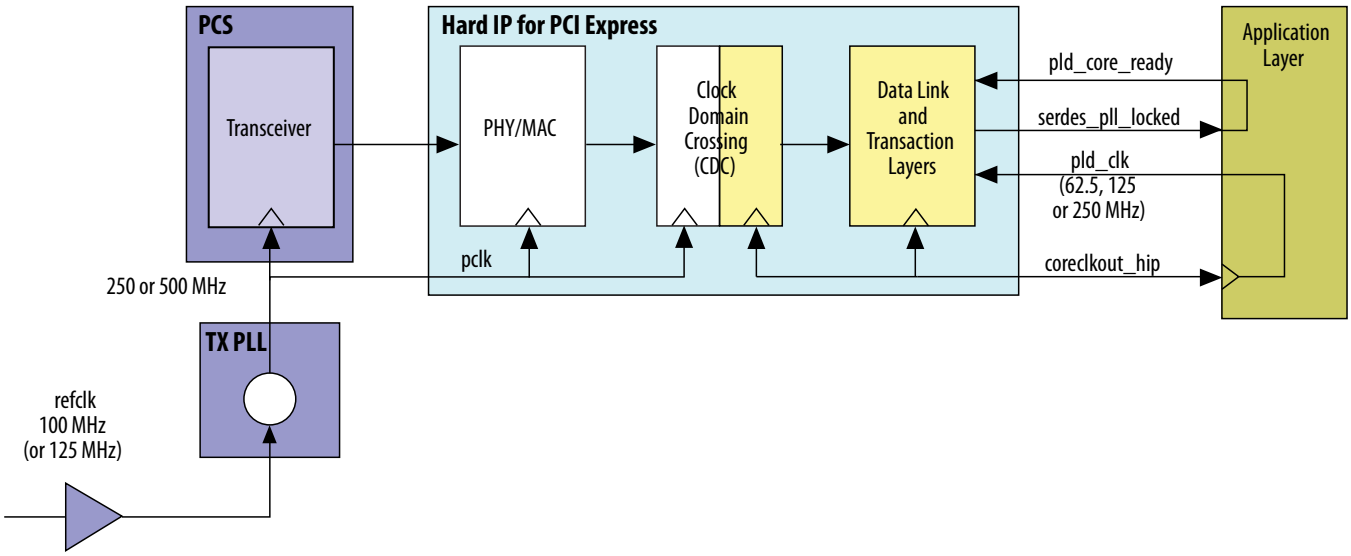
Related Information

PCI Express Base Specification 2.1 or 3.0

Clock Domains

Figure 6-5: Clock Domains and Clock Generation for the Application Layer

The following illustrates the clock domains when using `coreclkout_hip` to drive the Application Layer and the `pld_clk` of the IP core. The Altera-provided example design connects `coreclkout_hip` to the `pld_clk`. However, this connection is not mandatory.



As this figure indicates, the IP core includes the following clock domains:

pclk

The transceiver derives `pclk` from the 100 MHz `refclk` signal that you must provide to the device.

The *PCI Express Base Specification* requires that the `refclk` signal frequency be 100 MHz ±300 PPM.

The transitions between Gen1 and Gen2 should be glitchless. `pclk` can be turned off for most of the 1 ms timeout assigned for the PHY to change the clock rate; however, `pclk` should be stable before the 1 ms timeout expires.

Table 6-1: pclk Clock Frequency

Data Rate	Frequency
Gen1	250 MHz
Gen2	500 MHz

The CDC module implements the asynchronous clock domain crossing between the PHY/MAC `pclk` domain and the Data Link Layer `coreclk` domain. The transceiver `pclk` clock is connected directly to the Hard IP for PCI Express and does not connect to the FPGA fabric.

Related Information

PCI Express Base Specification 2.1 or 3.0

coreclkout_hip

Table 6-2: Application Layer Clock Frequency for All Combinations of Link Width, Data Rate and Application Layer Interface Widths

The `coreclkout_hip` signal is derived from `pclk`. The following table lists frequencies for `coreclkout_hip`, which are a function of the link width, data rate, and the width of the Application Layer to Transaction Layer interface. The frequencies and widths specified in this table are maintained throughout operation. If the link downtrains to a lesser link width or changes to a different maximum link rate, it maintains the frequencies it was originally configured for as specified in this table. (The Hard IP throttles the interface to achieve a lower throughput.)

Link Width	Maximum Link Rate	Avalon Interface Width	coreclkout_hip
×1	Gen1	64	62.5 MHz(4)
×1	Gen1	64	125 MHz
×2	Gen1	64	125 MHz
×4	Gen1	64	125 MHz
×8	Gen1	128	125 MHz
×1	Gen2	64	125 MHz

(4) This mode saves power

Link Width	Maximum Link Rate	Avalon Interface Width	coreclkout_hip
×2	Gen2	64	125 MHz
×4	Gen2	128	125 MHz

pld_clk

`coreclkout_hip` can drive the Application Layer clock along with the `pld_clk` input to the IP core. The `pld_clk` can optionally be sourced by a different clock than `coreclkout_hip`. The `pld_clk` minimum frequency cannot be lower than the `coreclkout_hip` frequency. Based on specific Application Layer constraints, a PLL can be used to derive the desired frequency.

Clock Summary

Table 6-3: Clock Summary

Name	Frequency	Clock Domain
`coreclkout_hip`	62.5, 125 or 250 MHz	Avalon-ST interface between the Transaction and Application Layers.
`pld_clk`	62.5, 125, or 250 MHz	Application and Transaction Layers.
`refclk`	100 or 125 MHz	SERDES (transceiver). Dedicated free running input clock to the SERDES block.
`reconfig_xcvr_clk`	100 –125 MHz	Transceiver Reconfiguration Controller.

Interrupts for Endpoints 7

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

The PCI Express Avalon-MM bridge supports MSI or legacy interrupts. The completer only single dword variant includes an interrupt handler that implements both INTX and MSI interrupts. Support requires instantiation of the CRA slave module where the interrupt registers and control logic are implemented.

The PCI Express Avalon-MM bridge supports the Avalon-MM individual requests interrupt scheme: multiple input signals indicate incoming interrupt requests, and software must determine priorities for servicing simultaneous interrupts.

The RX master module port has up to 16 Avalon-MM interrupt input signals (`RXmirq_irq[ <n> :0]`, where <*n*> ≤15). Each interrupt signal indicates a distinct interrupt source. Assertion of any of these signals, or a PCI Express mailbox register write access, sets a bit in the `Avalon-MM to PCI Express Interrupt Status` register. Multiple bits can be set at the same time; Application Layer software on the host side determines priorities for servicing simultaneous incoming interrupt requests. Each set bit in the `Avalon-MM to PCI Express Interrupt Status` register generates a PCI Express interrupt, if enabled, when software determines its turn. Software can enable the individual interrupts by writing to the `Avalon-MM to PCI Express Interrupt Enable Register` through the CRA slave.

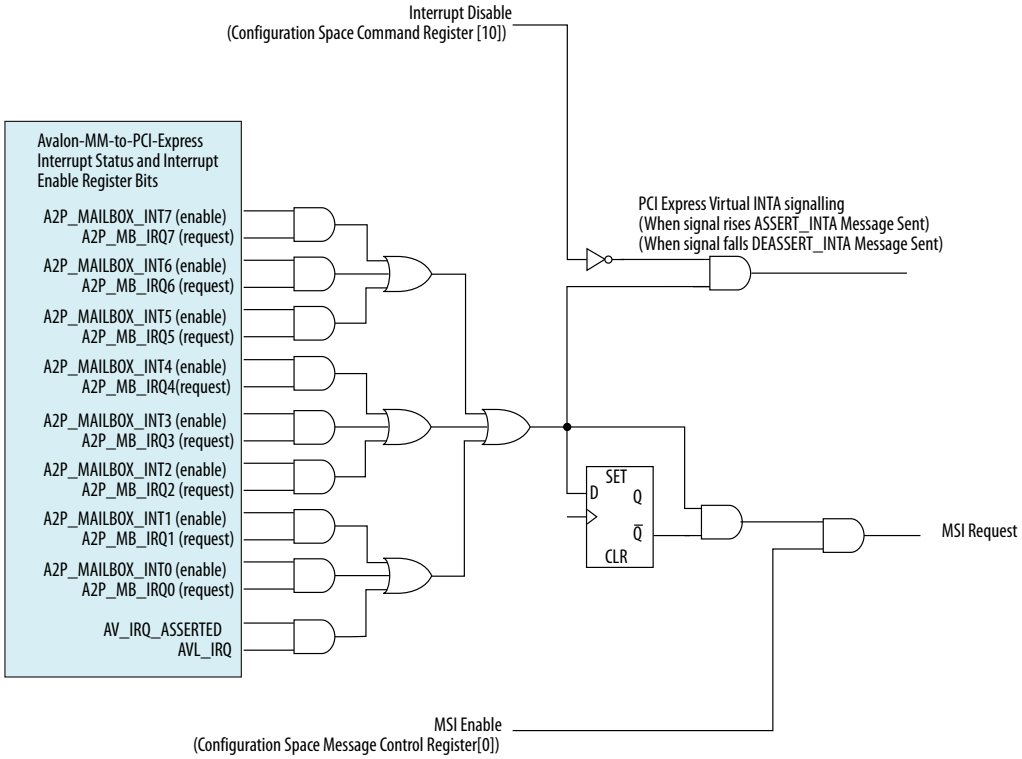
When any interrupt input signal is asserted, the corresponding bit is written in the `Avalon-MM to PCI Express Interrupt Status Register`. Software reads this register and decides priority on servicing requested interrupts.

After servicing the interrupt, software must clear the appropriate serviced interrupt `status` bit and ensure that no other interrupts are pending. For interrupts caused by `Avalon-MM to PCI Express Interrupt Status Register` mailbox writes, the status bits should be cleared in the `Avalon-MM to PCI Express Interrupt Status Register`. For interrupts due to the incoming interrupt signals on the Avalon-MM interface, the interrupt status should be cleared in the Avalon-MM component that sourced the interrupt. This sequence prevents interrupt requests from being lost during interrupt servicing.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 7-1: Avalon-MM Interrupt Propagation to the PCI Express Link



Related Information

- **Avalon-MM to PCI Express Interrupt Enable Registers** on page 5-16
- **Avalon-MM to PCI Express Interrupt Status Registers** on page 5-15

Enabling MSI or Legacy Interrupts

The PCI Express Avalon-MM bridge selects either MSI or legacy interrupts automatically based on the standard interrupt controls in the PCI Express Configuration Space registers. Software can write the `Interrupt Disable` bit, which is bit 10 of the `Command` register (at Configuration Space offset 0x4) to disable legacy interrupts. Software can write the `MSI Enable` bit, which is bit 0 of the `MSI Control Status` register in the MSI capability register (bit 16 at configuration space offset 0x50), to enable MSI interrupts.

Software can only enable one type of interrupt at a time. However, to change the selection of MSI or legacy interrupts during operation, software must ensure that no interrupt request is dropped. Therefore, software must first enable the new selection and then disable the old selection. To set up legacy interrupts, software must first clear the `Interrupt Disable` bit and then clear the `MSI enable` bit. To set up MSI interrupts, software must first set the `MSI enable` bit and then set the `Interrupt Disable` bit.

Generation of Avalon-MM Interrupts

The generation of Avalon-MM interrupts requires the instantiation of the CRA slave module where the interrupt registers and control logic are implemented. The CRA slave port has an Avalon-MM Interrupt output signal, `cra_Irq_irq`. A write access to an Avalon-MM mailbox register sets one of the `P2A_MAILBOX_INT<n>` bits in the `Avalon-MM to PCI Express Interrupt Status Register` and asserts the `cra_Irq_o` or `cra_Irq_irq` output, if enabled. Software can enable the interrupt by writing to the `INT_X Interrupt Enable Register for Endpoints` through the CRA slave. After servicing the interrupt, software must clear the appropriate serviced interrupt `status` bit in the PCI-Express-to-Avalon-MM `Interrupt Status` register and ensure that no other interrupt is pending.

Related Information

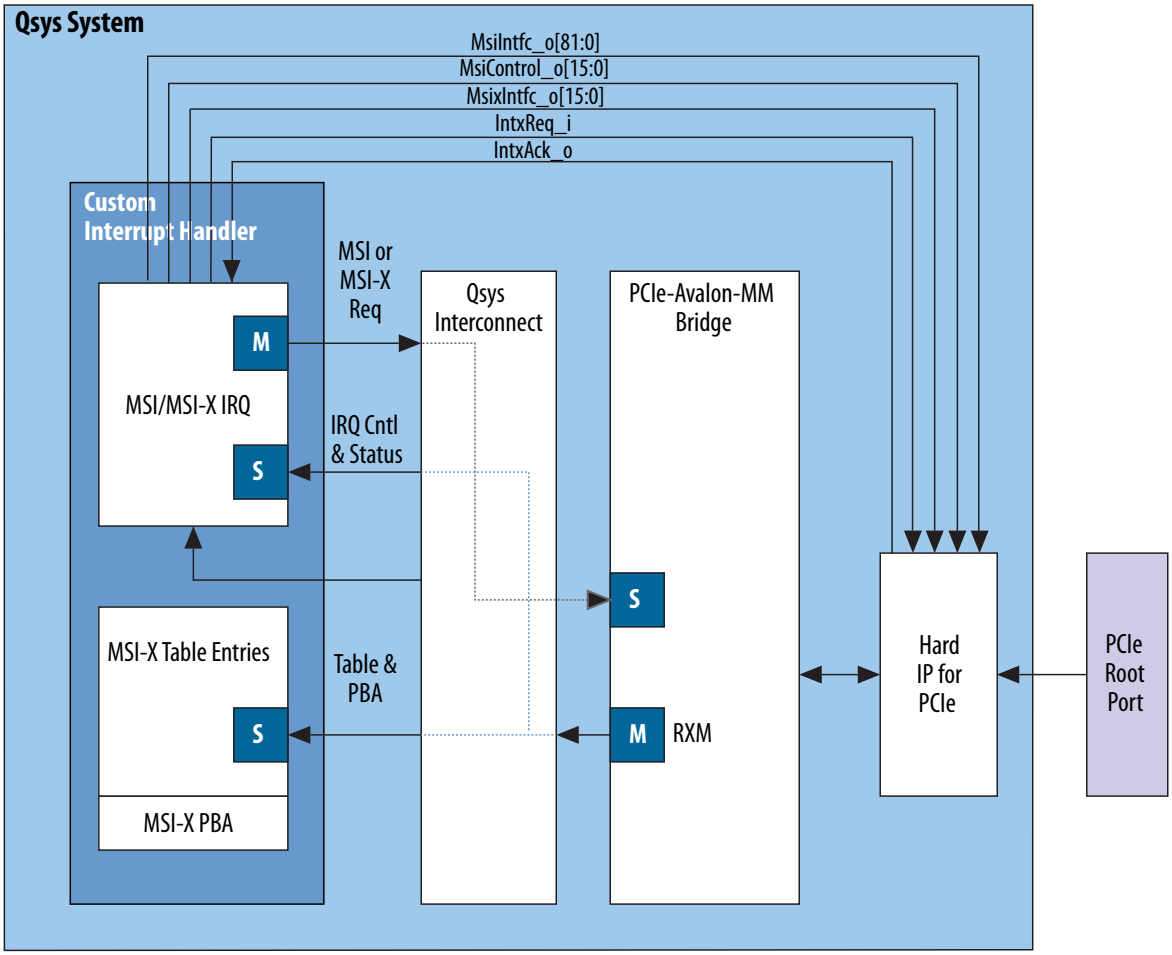
- **Avalon-MM to PCI Express Interrupt Status Registers** on page 5-15
- **PCI Express to Avalon-MM Interrupt Status and Enable Registers for Endpoints** on page 5-19

Interrupts for Endpoints Using the Avalon-MM Interface with Multiple MSI/MSI-X Support

If you select **Enable multiple MSI/MSI X support** under the **Avalon-MM System Settings** banner in the parameter editor, the Hard IP for PCI Express exports the MSI, MSI-X, and INTx interfaces to the Application Layer. The Application Layer must include a Custom Interrupt Handler to send interrupts to the Root Port. You must design this Custom Interrupt Handler. The following figure provides an overview of the logic for the Custom Interrupt Handler. The Custom Interrupt Handler should include hardware to perform the following tasks:

- An MSI/MXI-X IRQ Avalon-MM Master port to drive MSI or MSI-X interrupts as memory writes to the PCIe Avalon-MM bridge.
- A legacy interrupt signal, `IntxReq_i`, to drive legacy interrupts from the MSI/MSI-X IRQ module to the Hard IP for PCI Express.
- An MSI/MSI-X Avalon-MM Slave port to receive interrupt control and status from the PCIe Root Port.
- An MSI-X table to store the MSI-X table entries. The PCIe Root Port sets up this table.

Figure 7-2: Block Diagram for Custom Interrupt Handler



Refer to *Interrupts for Endpoints* for the definitions of MSI, MSI-X, and INTx buses.

For more information about implementing MSI or MSI-X interrupts, refer to the *PCI Local Bus Specification, Revision 2.3, MSI-X ECN*.

For more information about implementing interrupts, including an MSI design example, refer to *Handling PCIe Interrupts* on the Altera wiki.

Related Information

- **Interrupts for Endpoints** on page 7-1
- **PCI Local Bus Specification, Revision 2.3**
- **Handling PCIe Interrupts**

Error Handling 8

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

Each PCI Express compliant device must implement a basic level of error management and can optionally implement advanced error management. The IP core implements both basic and advanced error reporting. Error handling for a Root Port is more complex than that of an Endpoint.

Table 8-1: Error Classification

The *PCI Express Base Specification* defines three types of errors, outlined in the following table.

Type	Responsible Agent	Description
Correctable	Hardware	While correctable errors may affect system performance, data integrity is maintained.
Uncorrectable, non-fatal	Device software	Uncorrectable, non-fatal errors are defined as errors in which data is lost, but system integrity is maintained. For example, the fabric may lose a particular TLP, but it still works without problems.
Uncorrectable, fatal	System software	Errors generated by a loss of data and system failure are considered uncorrectable and fatal. Software must determine how to handle such errors: whether to reset the link or implement other means to minimize the problem.

Related Information

PCI Express Base Specification 2.1 and 3.0

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Physical Layer Errors

Table 8-2: Errors Detected by the Physical Layer

The following table describes errors detected by the Physical Layer. Physical Layer error reporting is optional in the *PCI Express Base Specification*.

Error	Type	Description

Receive port error	Correctable	This error has the following 3 potential causes:
		• Physical coding sublayer error when a lane is in L0 state. These errors are reported to the Hard IP block via the per lane PIPE interface input receive status signals, `rxstatus<lane_number>[2:0]` using the following encodings:
		  • 3'b100: 8B/10B Decode Error
		  • 3'b101: Elastic Buffer Overflow
		  • 3'b110: Elastic Buffer Underflow
		  • 3'b111: Disparity Error
		• Deskew error caused by overflow of the multilane deskew FIFO.
		• Control symbol received in wrong lane.

Data Link Layer Errors

Table 8-3: Errors Detected by the Data Link Layer

Error	Type	Description
Bad TLP	Correctable	This error occurs when a LCRC verification fails or when a sequence number error occurs.
Bad DLLP	Correctable	This error occurs when a CRC verification fails.
Replay timer	Correctable	This error occurs when the replay timer times out.
Replay num rollover	Correctable	This error occurs when the replay number rolls over.
Data Link Layer protocol	Uncorrectable(fatal)	This error occurs when a sequence number specified by the Ack/Nak block in the Data Link Layer (`AckNak_Seq_Num`) does not correspond to an unacknowledged TLP.

Transaction Layer Errors

Table 8-4: Errors Detected by the Transaction Layer

Error	Type	Description
Poisoned TLP received	Uncorrectable (non-fatal)	This error occurs if a received Transaction Layer packet has the EP poison bit set. The received TLP is passed to the Application Layer and the Application Layer logic must take appropriate action in response to the poisoned TLP. Refer to "2.7.2.2 Rules for Use of Data Poisoning" in the *PCI Express Base Specification* for more information about poisoned TLPs.
ECRC check failed (1)	Uncorrectable (non-fatal)	This error is caused by an ECRC check failing despite the fact that the TLP is not malformed and the LCRC check is valid. The Hard IP block handles this TLP automatically. If the TLP is a non-posted request, the Hard IP block generates a completion with completer abort status. In all cases the TLP is deleted in the Hard IP block and not presented to the Application Layer.
Unsupported Request for Endpoints	Uncorrectable (non-fatal)	This error occurs whenever a component receives any of the following Unsupported Requests: • Type 0 Configuration Requests for a non-existing function. • Completion transaction for which the Requester ID does not match the bus, device and function number. • Unsupported message. • A Type 1 Configuration Request TLP for the TLP from the PCIe link. • A locked memory read (MEMRDLK) on native Endpoint. • A locked completion transaction. • A 64-bit memory transaction in which the 32 MSBs of an address are set to 0. • A memory or I/O transaction for which there is no BAR match. • A memory transaction when the Memory Space Enable bit (bit [1] of the PCI Command register at Configuration Space offset 0x4) is set to 0. • A poisoned configuration write request (`CfgWr0`)

Error	Type	Description
		In all cases the TLP is deleted in the Hard IP block and not presented to the Application Layer. If the TLP is a non-posted request, the Hard IP block generates a completion with Unsupported Request status.
Unsupported Requests for Root Port	Uncorrectable (fatal)	This error occurs whenever a component receives an Unsupported Request including: • Unsupported message • A Type 0 Configuration Request TLP • A 64-bit memory transaction which the 32 MSBs of an address are set to 0. • A memory transaction that does not match the address range defined by the Base and Limit Address registers
Completion timeout	Uncorrectable (non-fatal)	This error occurs when a request originating from the Application Layer does not generate a corresponding completion TLP within the established time. It is the responsibility of the Application Layer logic to provide the completion timeout mechanism. The completion timeout should be reported from the Transaction Layer using the `cpl_err[0]` signal.
Completer abort *(1)*	Uncorrectable (non-fatal)	The Application Layer reports this error using the `cpl_err[2]`signal when it aborts receipt of a TLP.

Error	Type	Description
Unexpected completion	Uncorrectable (non-fatal)	This error is caused by an unexpected completion transaction. The Hard IP block handles the following conditions: • The Requester ID in the completion packet does not match the Configured ID of the Endpoint. • The completion packet has an invalid tag number. (Typically, the tag used in the completion packet exceeds the number of tags specified.) • The completion packet has a tag that does not match an outstanding request. • The completion packet for a request that was to I/O or Configuration Space has a length greater than 1 dword. • The completion status is Configuration Retry Status (CRS) in response to a request that was not to Configuration Space. In all of the above cases, the TLP is not presented to the Application Layer; the Hard IP block deletes it. The Application Layer can detect and report other unexpected completion conditions using the `cpl_err[2]` signal. For example, the Application Layer can report cases where the total length of the received successful completions do not match the original read request length.
Receiver overflow *(1)*	Uncorrectable (fatal)	This error occurs when a component receives a TLP that violates the FC credits allocated for this type of TLP. In all cases the hard IP block deletes the TLP and it is not presented to the Application Layer.
Flow control protocol error (FCPE) *(1)*	Uncorrectable (fatal)	This error occurs when a component does not receive update flow control credits with the 200 µs limit.

Error	Type	Description
Malformed TLP	Uncorrectable (fatal)	This error is caused by any of the following conditions: • The data payload of a received TLP exceeds the maximum payload size. • The TD field is asserted but no TLP digest exists, or a TLP digest exists but the TD bit of the PCI Express request header packet is not asserted. • A TLP violates a byte enable rule. The Hard IP block checks for this violation, which is considered optional by the PCI Express specifications. • A TLP in which the type and length fields do not correspond with the total length of the TLP. • A TLP in which the combination of format and type is not specified by the PCI Express specification. • A request specifies an address/length combination that causes a memory space access to exceed a 4 KByte boundary. The Hard IP block checks for this violation, which is considered optional by the PCI Express specification. • Messages, such as Assert_INTX, Power Management, Error Signaling, Unlock, and Set Power Slot Limit, must be transmitted across the default traffic class. The Hard IP block deletes the malformed TLP; it is not presented to the Application Layer.

Note:

1. Considered optional by the *PCI Express Base Specification Revision* .

Error Reporting and Data Poisoning

How the Endpoint handles a particular error depends on the configuration registers of the device.

Refer to the *PCI Express Base Specification 3.0* for a description of the device signaling and logging for an Endpoint.

The Hard IP block implements data poisoning, a mechanism for indicating that the data associated with a transaction is corrupted. Poisoned TLPs have the error/poisoned bit of the header set to 1 and observe the following rules:

- Received poisoned TLPs are sent to the Application Layer and status bits are automatically updated in the Configuration Space.
- Received poisoned Configuration Write TLPs are not written in the Configuration Space.
- The Configuration Space never generates a poisoned TLP; the error/poisoned bit of the header is always set to 0.

Poisoned TLPs can also set the parity error bits in the PCI Configuration Space Status register.

Table 8-5: Parity Error Conditions

Status Bit	Conditions
Detected parity error (status register bit 15)	Set when any received TLP is poisoned.
Master data parity error (status register bit 8)	This bit is set when the command register parity enable bit is set and one of the following conditions is true: • The poisoned bit is set during the transmission of a Write Request TLP. • The poisoned bit is set on a received completion TLP.

Poisoned packets received by the Hard IP block are passed to the Application Layer. Poisoned transmit TLPs are similarly sent to the link.

Related Information

PCI Express Base Specification 2.1 and 3.0

Uncorrectable and Correctable Error Status Bits

The following section is reprinted with the permission of PCI-SIG. Copyright 2010 PCI-SIG.

Figure 8-1: Uncorrectable Error Status Register

The default value of all the bits of this register is 0. An error status bit that is set indicates that the error condition it represents has been detected. Software may clear the error status by writing a 1 to the appropriate bit.

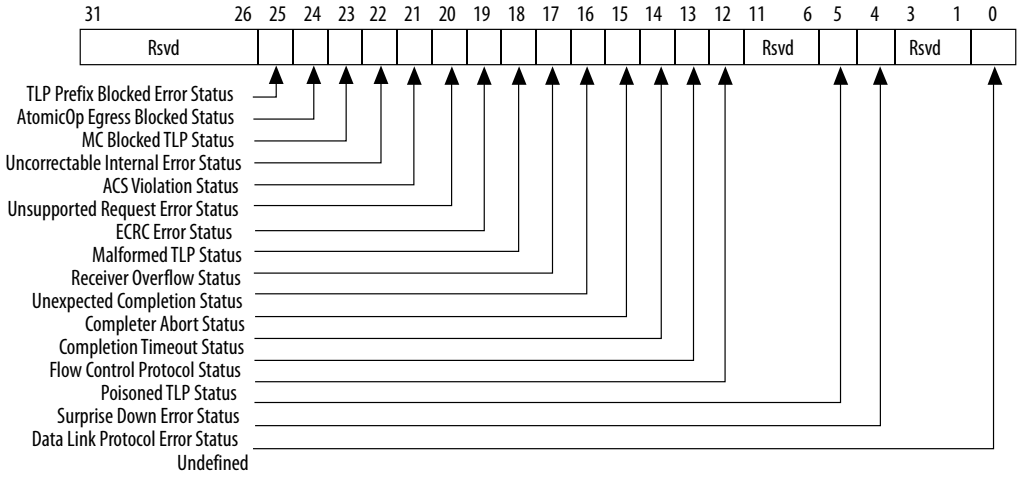
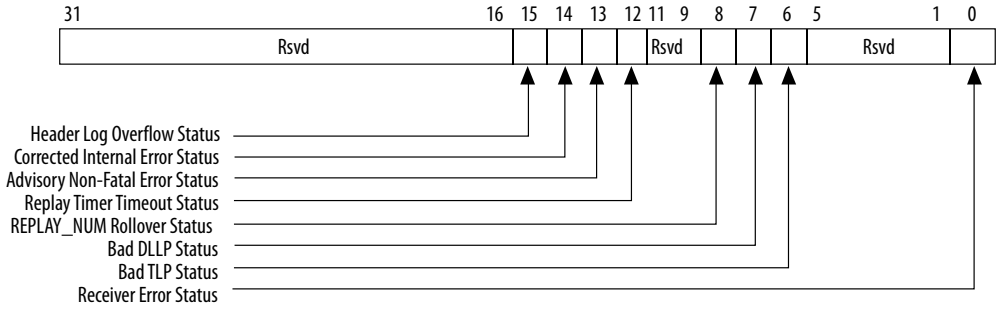


Figure 8-2: Correctable Error Status Register

The default value of all the bits of this register is 0. An error status bit that is set indicates that the error condition it represents has been detected. Software may clear the error status by writing a 1 to the appropriate bit.



IP Core Architecture 9

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

The Avalon-MM Arria V Hard IP for PCI Express implements the complete PCI Express protocol stack as defined in the *PCI Express Base Specification.* The protocol stack includes the following layers:

- *Transaction Layer*—The Transaction Layer contains the Configuration Space, which manages communication with the Application Layer, the RX and TX channels, the RX buffer, and flow control credits.
- *Data Link Layer*—The Data Link Layer, located between the Physical Layer and the Transaction Layer, manages packet transmission and maintains data integrity at the link level. Specifically, the Data Link Layer performs the following tasks:
  - Manages transmission and reception of Data Link Layer Packets (DLLPs)
  - Generates all transmission cyclical redundancy code (CRC) values and checks all CRCs during reception
  - Manages the retry buffer and retry mechanism according to received ACK/NAK Data Link Layer packets
  - Initializes the flow control mechanism for DLLPs and routes flow control credits to and from the Transaction Layer
- *Physical Layer*—The Physical Layer initializes the speed, lane numbering, and lane width of the PCI Express link according to packets received from the link and directives received from higher layers.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 9-1: Arria V Hard IP for PCI Express Using the Avalon-MM Interface

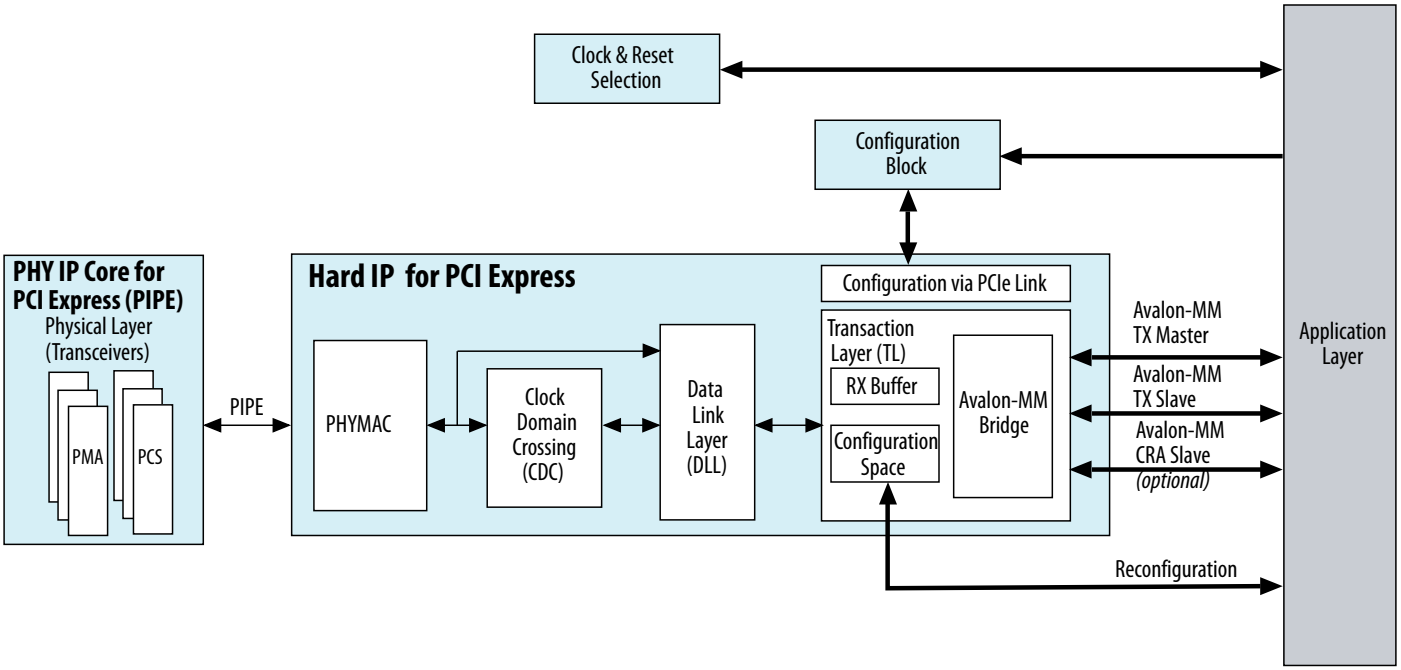


Table 9-1: Application Layer Clock Frequencies

Lanes	Gen1	Gen2
×1	125 MHz @ 64 bits or 62.5 MHz @ 64 bits	125 MHz @ 64 bits
×2	125 MHz @ 64 bits	125 MHz @ 64 bits
×4	125 MHz @ 64 bits	125 MHz @ 128 bits
×8	125 MHz @ 128 bits	N/A

Related Information

PCI Express Base Specification 2.1 or 3.0

Top-Level Interfaces

Avalon-MM Interface

An Avalon-MM interface connects the Application Layer and the Transaction Layer. The Avalon-MM interface implement the Avalon-MM protocol described in the *Avalon Interface Specifications*. Refer to this specification for information about the Avalon-MM protocol, including timing diagrams.

Related Information

- **64- or 128-Bit Avalon-MM Interface to the Application Layer** on page 4-1
- **Avalon Interface Specifications**

Clocks and Reset

The *PCI Express Base Specification* requires an input reference clock, which is called `refclk` in this design. The *PCI Express Base Specification* stipulates that the frequency of this clock be 100 MHz.

The *PCI Express Base Specification* also requires a system configuration time of 100 ms. To meet this specification, IP core includes an embedded hard reset controller. This reset controller exits the reset state after the I/O ring of the device is initialized.

Transceiver Reconfiguration

The transceiver reconfiguration interface allows you to dynamically reconfigure the values of analog settings in the PMA block of the transceiver. Dynamic reconfiguration is necessary to compensate for process variations.

Related Information

Transceiver PHY IP Reconfiguration on page 12-1

Interrupts

The Hard IP for PCI Express offers the following interrupt mechanisms:

- Message Signaled Interrupts (MSI)— MSI uses the Transaction Layer's request-acknowledge handshaking protocol to implement interrupts. The MSI Capability structure is stored in the Configuration Space and is programmable using Configuration Space accesses.
- MSI-X—The Transaction Layer generates MSI-X messages which are single dword memory writes. In contrast to the MSI capability structure, which contains all of the control and status information for the interrupt vectors, the MSI-X Capability structure points to an MSI-X table structure and MSI-X PBA structure which are stored in memory.

Related Information

Interrupts for Endpoints when Multiple MSI/MSI-X Support Is Enabled on page 4-13

PIPE

The PIPE interface implements the Intel-designed PIPE interface specification. You can use this parallel interface to speed simulation; however, you cannot use the PIPE interface in actual hardware.

- The Gen1, Gen2, and Gen3 simulation models support PIPE and serial simulation.
- For Gen3, the Altera BFM bypasses Gen3 Phase 2 and Phase 3 Equalization. However, Gen3 variants can perform Phase 2 and Phase 3 equalization if instructed by a third-party BFM.

Related Information

PIPE Interface Signals

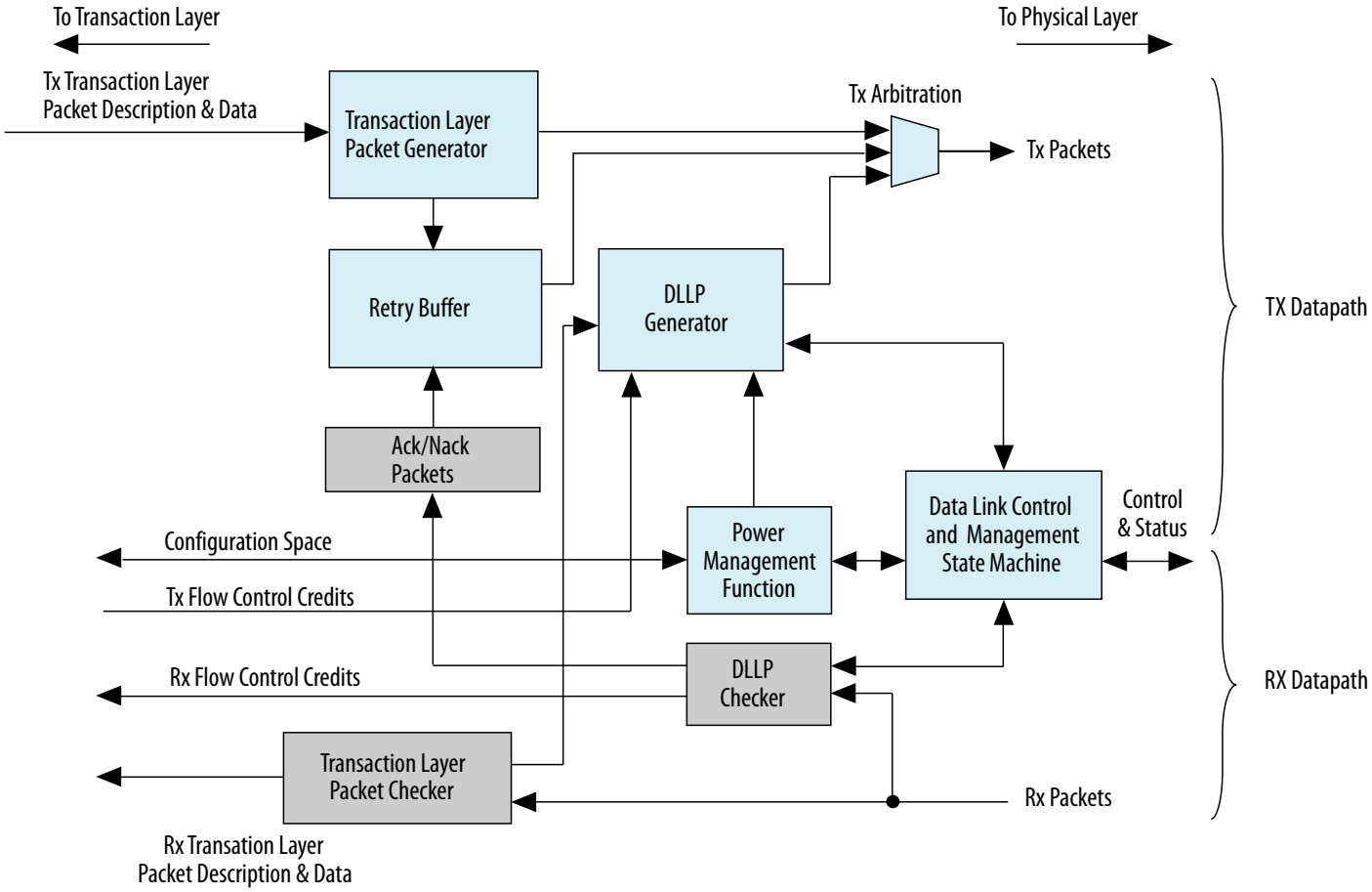
Data Link Layer

The Data Link Layer is located between the Transaction Layer and the Physical Layer. It maintains packet integrity and communicates (by DLL packet transmission) at the PCI Express link level (as opposed to component communication by TLP transmission in the interconnect fabric).

The DLL implements the following functions:

- Link management through the reception and transmission of DLL packets (DLLP), which are used for the following functions:
  - Power management of DLLP reception and transmission
  - To transmit and receive ACK/NACK packets
  - Data integrity through generation and checking of CRCs for TLPs and DLLPs
  - TLP retransmission in case of NAK DLLP reception using the retry buffer
  - Management of the retry buffer
  - Link retraining requests in case of error through the Link Training and Status State Machine (LTSSM) of the Physical Layer

Figure 9-2: Data Link Layer



The DLL has the following sub-blocks:

- Data Link Control and Management State Machine—This state machine is synchronized with the Physical Layer's LTSSM state machine and is also connected to the Configuration Space Registers. It initializes the link and flow control credits and reports status to the Configuration Space.
- Power Management—This function handles the handshake to enter low power mode. Such a transition is based on register values in the Configuration Space and received Power Management (PM) DLLPs.
- Data Link Layer Packet Generator and Checker—This block is associated with the DLLP's 16-bit CRC and maintains the integrity of transmitted packets.
- Transaction Layer Packet Generator—This block generates transmit packets, generating a sequence number and a 32-bit CRC (LCRC). The packets are also sent to the retry buffer for internal storage. In retry mode, the TLP generator receives the packets from the retry buffer and generates the CRC for the transmit packet.
- Retry Buffer—The retry buffer stores TLPs and retransmits all unacknowledged packets in the case of NAK DLLP reception. In case of ACK DLLP reception, the retry buffer discards all acknowledged packets.

- ACK/NAK Packets—The ACK/NAK block handles ACK/NAK DLLPs and generates the sequence number of transmitted packets.
- Transaction Layer Packet Checker—This block checks the integrity of the received TLP and generates a request for transmission of an ACK/NAK DLLP.
- TX Arbitration—This block arbitrates transactions, prioritizing in the following order:
  - Initialize FC Data Link Layer packet
  - ACK/NAK DLLP (high priority)
  - Update FC DLLP (high priority)
  - PM DLLP
  - Retry buffer TLP
  - TLP
  - Update FC DLLP (low priority)
  - ACK/NAK FC DLLP (low priority)

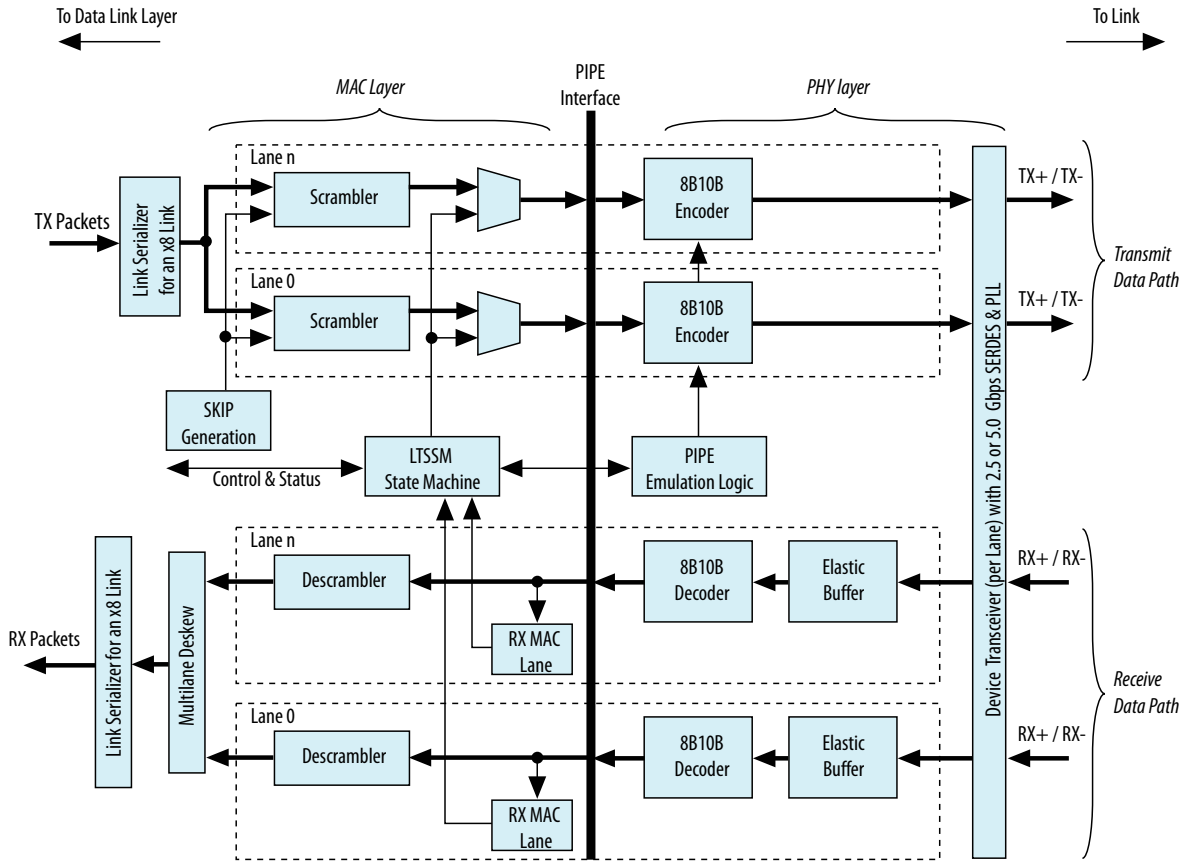
Physical Layer

The Physical Layer is the lowest level of the PCI Express protocol stack. It is the layer closest to the serial link. It encodes and transmits packets across a link and accepts and decodes received packets. The Physical Layer connects to the link through a high-speed SERDES interface running at 2.5 Gbps for Gen1 implementations and a5 2.5 or 5.0 Gbps for Gen2 implementations.

The Physical Layer is responsible for the following actions:

- Initializing the link
- Scrambling/descrambling and 8B/10B encoding/decoding for 2.5 Gbps (Gen1) and 5.0 Gbps (Gen2) per lane
- Serializing and deserializing data
- Operating the PIPE 3.0 Interface
- Implementing auto speed negotiation (Gen2 and Gen3)
- Transmitting and decoding the training sequence
- Providing hardware autonomous speed control
- Implementing auto lane reversal

Figure 9-3: Physical Layer Architecture



The Physical Layer is subdivided by the PIPE Interface Specification into two layers (bracketed horizontally in above figure):

- Media Access Controller (MAC) Layer—The MAC layer includes the LTSSM and the scrambling/descrambling and multilane deskew functions.
- PHY Layer—The PHY layer includes the 8B/10B and 128b/130b encode/decode functions, elastic buffering, and serialization/deserialization functions.

The Physical Layer integrates both digital and analog elements. Intel designed the PIPE interface to separate the MAC from the PHY. The Arria V Hard IP for PCI Express complies with the PIPE interface specification.

The PHYMAC block comprises four main sub-blocks:

- MAC Lane—Both the RX and the TX path use this block.
  - On the RX side, the block decodes the Physical Layer packet and reports to the LTSSM the type and number of TS1/TS2 ordered sets received.
  - On the TX side, the block multiplexes data from the DLL and the LTSTX sub-block. It also adds lane specific information, including the lane number and the force PAD value when the LTSSM disables the lane during initialization.
- LTSSM—This block implements the LTSSM and logic that tracks TX and RX data on each lane.
- For transmission, it interacts with each MAC lane sub-block and with the LTSTX sub-block by asserting both global and per-lane control bits to generate specific Physical Layer packets.
  - On the receive path, it receives the Physical Layer packets reported by each MAC lane sub-block. It also enables the multilane deskew block. This block reports the Physical Layer status to higher layers.
  - LTSTX (Ordered Set and SKP Generation)—This sub-block generates the Physical Layer packet. It receives control signals from the LTSSM block and generates Physical Layer packet for each lane. It generates the same Physical Layer Packet for all lanes and PAD symbols for the link or lane number in the corresponding TS1/TS2 fields.

The block also handles the receiver detection operation to the PCS sub-layer by asserting predefined PIPE signals and waiting for the result. It also generates a SKP Ordered Set at every predefined timeslot and interacts with the TX alignment block to prevent the insertion of a SKP Ordered Set in the middle of packet.

- Deskew—This sub-block performs the multilane deskew function and the RX alignment between the number of initialized lanes and the 64-bit data path.

The multilane deskew implements an eight-word FIFO buffer for each lane to store symbols. Each symbol includes eight data bits, one disparity bit, and one control bit. The FIFO discards the FTS, COM, and SKP symbols and replaces PAD and IDL with D0.0 data. When all eight FIFOs contain data, a read can occur.

When the multilane lane deskew block is first enabled, each FIFO begins writing after the first COM is detected. If all lanes have not detected a COM symbol after seven clock cycles, they are reset and the resynchronization process restarts, or else the RX alignment function recreates a 64-bit data word which is sent to the DLL.

32-Bit PCI Express Avalon-MM Bridge

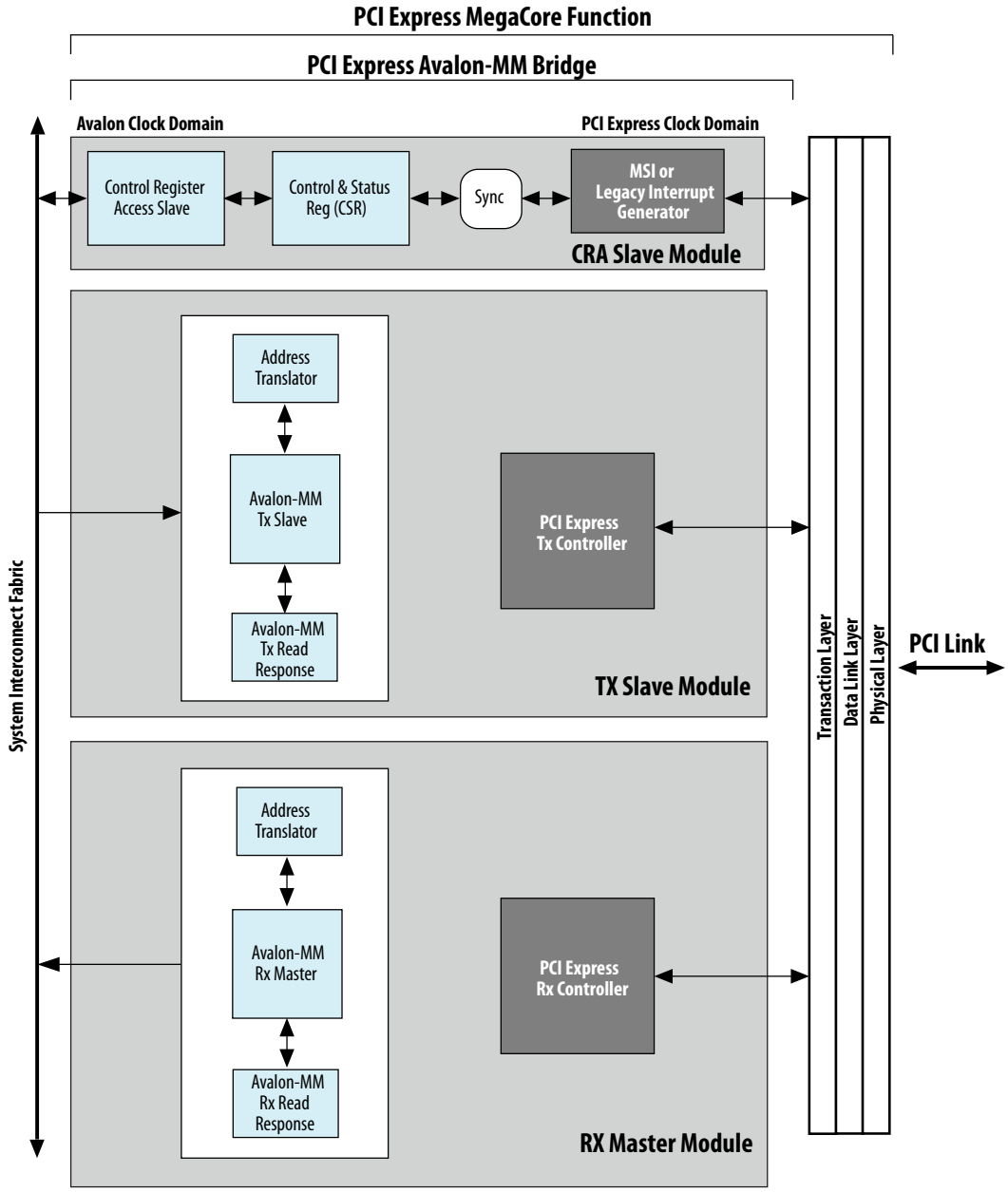
The Avalon-MM Arria V Hard IP for PCI Express includes an Avalon-MM bridge module that connects the Hard IP to the interconnect fabric. The bridge facilitates the design of Endpoints and Root Ports that include Qsys components.

The Avalon-MM bridge provides three possible Avalon-MM ports: a bursting master, an optional bursting slave, and an optional non-bursting slave. The Avalon-MM bridge comprises the following three modules:

- TX Slave Module—This optional 64- or 128-bit bursting, Avalon-MM dynamic addressing slave port propagates read and write requests of up to 4 KBytes in size from the interconnect fabric to the PCI Express link. The bridge translates requests from the interconnect fabric to PCI Express request packets.
- RX Master Module—This 64- or 128-bit bursting Avalon-MM master port propagates PCI Express requests, converting them to bursting read or write requests to the interconnect fabric.
- Control Register Access (CRA) Slave Module—This optional, 32-bit Avalon-MM dynamic addressing slave port provides access to internal control and status registers from upstream PCI Express devices and external Avalon-MM masters. Implementations that use MSI or dynamic address translation require this port. The CRA port supports single dword read and write requests. It does not support bursting.

When you select the **Single dword completer** for the Avalon-MM Hard IP for PCI Express, Qsys substitutes a unpipelined, 32-bit RX master port for the 64- or 128-bit full-featured RX master port. The following figure shows the block diagram of a full-featured PCI Express Avalon-MM bridge.

Figure 9-4: PCI Express Avalon-MM Bridge



The bridge has the following additional characteristics:

- Type 0 and Type 1 vendor-defined incoming messages are discarded
- Completion-to-a-flush request is generated, but not propagated to the interconnect fabric

For End Points, each PCI Express base address register (BAR) in the Transaction Layer maps to a specific, fixed Avalon-MM address range. You can use separate BARs to map to various Avalon-MM slaves connected to the RX Master port. In contrast to Endpoints, Root Ports do not perform any BAR matching and forwards the address to a single RX Avalon-MM master port.

Related Information

Avalon-MM RX Master Block on page 9-20

Avalon-MM Bridge TLPs

The PCI Express to Avalon-MM bridge translates the PCI Express read, write, and completion Transaction Layer Packets (TLPs) into standard Avalon-MM read and write commands typically used by master and slave interfaces. This PCI Express to Avalon-MM bridge also translates Avalon-MM read, write and read data commands to PCI Express read, write and completion TLPs. The following topics describe the Avalon-MM bridges translations.

Avalon-MM-to-PCI Express Write Requests

The Avalon-MM bridge accepts Avalon-MM burst write requests with a burst size of up to 512 Bytes at the Avalon-MM TX slave interface. The Avalon-MM bridge converts the write requests to one or more PCI Express write packets with 32– or 64-bit addresses based on the address translation configuration, the request address, and the maximum payload size.

The Avalon-MM write requests can start on any address in the range defined in the PCI Express address table parameters. The bridge splits incoming burst writes that cross a 4 KByte boundary into at least two separate PCI Express packets. The bridge also considers the root complex requirement for maximum payload on the PCI Express side by further segmenting the packets if needed.

The bridge requires Avalon-MM write requests with a burst count of greater than one to adhere to the following byte enable rules:

- The Avalon-MM byte enables must be asserted in the first qword of the burst.
- All subsequent byte enables must be asserted until the deasserting byte enable.
- The Avalon-MM byte enables may deassert, but only in the last qword of the burst.

Note: To improve PCI Express throughput, Altera recommends using an Avalon-MM burst master without any byte-enable restrictions.

Avalon-MM-to-PCI Express Upstream Read Requests

The PCI Express Avalon-MM bridge converts read requests from the system interconnect fabric to PCI Express read requests with 32-bit or 64-bit addresses based on the address translation configuration, the request address, and the maximum read size.

The Avalon-MM TX slave interface of a PCI Express Avalon-MM bridge can receive read requests with burst sizes of up to 512 bytes sent to any address. However, the bridge limits read requests sent to the PCI Express link to a maximum of 256 bytes. Additionally, the bridge must prevent each PCI Express read request packet from crossing a 4 KByte address boundary. Therefore, the bridge may split an Avalon-MM read request into multiple PCI Express read packets based on the address and the size of the read request.

Avalon-MM bridge supports up to eight outstanding reads from Avalon-MM interface. Once the bridge has eight outstanding read requests, the `txs_waitrequest` signal is asserted to block additional read requests. When a read request completes, the Avalon-MM bridge can accept another request.

For Avalon-MM read requests with a burst count greater than one, all byte enables must be asserted. There are no restrictions on byte enables for Avalon-MM read requests with a burst count of one. An invalid Avalon-MM request can adversely affect system functionality, resulting in a completion with the abort status set. An example of an invalid request is one with an incorrect address.

PCI Express-to-Avalon-MM Read Completions

The PCI Express Avalon-MM bridge returns read completion packets to the initiating Avalon-MM master in the issuing order. The bridge supports multiple and out-of-order completion packets.

PCI Express-to-Avalon-MM Downstream Write Requests

The PCI Express Avalon-MM bridge receives PCI Express write requests, it converts them to burst write requests before sending them to the interconnect fabric. For Endpoints, the bridge translates the PCI Express address to the Avalon-MM address space based on the BAR hit information and on address translation table values configured during the IP core parameterization. For Root Ports, all requests are forwarded to a single RX Avalon-MM master that drives them to the interconnect fabric. Malformed write packets are dropped, and therefore do not appear on the Avalon-MM interface.

For downstream write and read requests, if more than one byte enable is asserted, the byte lanes must be adjacent. In addition, the byte enables must be aligned to the size of the read or write request.

As an example, the following table lists the byte enables for 32-bit data.

Table 9-2: Valid Byte Enable Configurations

Byte Enable Value	Description
4'b1111	Write full 32 bits
4'b0011	Write the lower 2 bytes
4'b1100	Write the upper 2 bytes
4'b0001	Write byte 0 only
4'b0010	Write byte 1 only
4'b0100	Write byte 2 only
4'b1000	Write byte 3 only

In burst mode, the Arria V Hard IP for PCI Express supports only byte enable values that correspond to a contiguous data burst. For the 32-bit data width example, valid values in the first data phase are 4'b1111, 4'b1110, 4'b1100, and 4'b1000, and valid values in the final data phase of the burst are 4'b1111, 4'b0111, 4'b0011, and 4'b0001. Intermediate data phases in the burst can only have byte enable value 4'b1111.

PCI Express-to-Avalon-MM Downstream Read Requests

The PCI Express Avalon-MM bridge sends PCI Express read packets to the interconnect fabric as burst reads with a maximum burst size of 512 bytes. For Endpoints, the bridge converts the PCI Express address to the Avalon-MM address space based on the BAR hit information and address translation lookup table values. The RX Avalon-MM master port drives the received address to the fabric. You can set up the Address Translation Table Configuration in the parameter editor. Unsupported read requests generate a completer abort response.

Related Information

Minimizing BAR Sizes and the PCIe Address Space on page 9-15

Avalon-MM-to-PCI Express Read Completions

The PCI Express Avalon-MM bridge converts read response data from Application Layer Avalon-MM slaves to PCI Express completion packets and sends them to the Transaction Layer.

A single read request may produce multiple completion packets based on the **Maximum payload size** and the size of the received read request. For example, if the read is 512 bytes but the **Maximum payload size** 128 bytes, the bridge produces four completion packets of 128 bytes each. The bridge does not generate out-of-order completions even to different BARs. You can specify the **Maximum payload size** parameter on the **Device** tab under the **PCI Express/PCI Capabilities** heading in the parameter editor.

Related Information

Device Capabilities

PCI Express-to-Avalon-MM Address Translation for 32-Bit Bridge

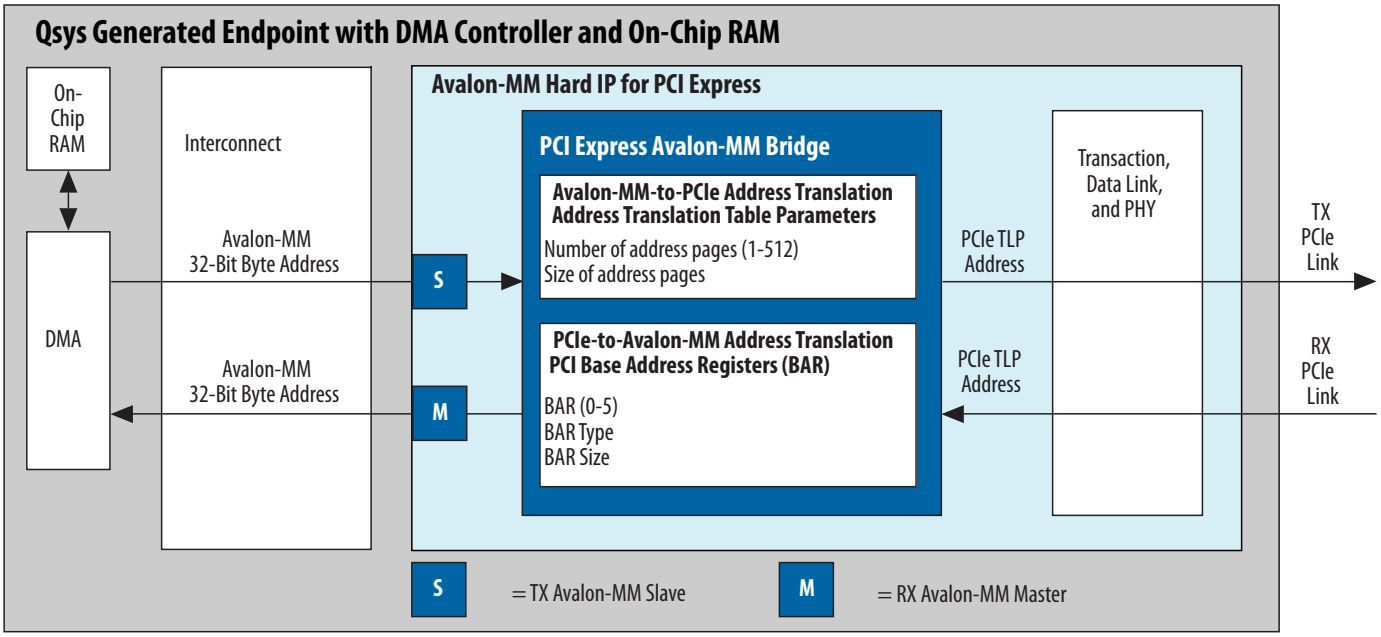
The PCI Express Avalon-MM bridge translates the system-level physical addresses, typically up to 64 bits, to the significantly smaller addresses required by the Application Layer's Avalon-MM slave components.

Note: Starting with the 13.0 version of the Quartus II software, the PCI Express-to-Avalon-MM bridge supports both 32- and 64-bit addresses. If you select 64-bit addressing the bridge does not perform address translation. It drives the addresses specified to the interconnect fabric. You can limit the number of address bits used by Avalon-MM slave components to the actual size required by specifying the address size in the Avalon-MM slave component parameter editor.

You can specify up to six BARs for address translation when you customize your Hard IP for PCI Express as described in *Base Address Register (BAR) and Expansion ROM Settings*. When 32-bit addresses are specified, the PCI Express Avalon-MM bridge also translates Application Layer addresses to system-level physical addresses as described in *Avalon-MM-to-PCI Express Address Translation Algorithm for 32-Bit Addressing*.

The following figure provides a high-level view of address translation in both directions.

Figure 9-5: Address Translation in TX and RX Directions For Endpoints



Note: When configured as a Root Port, a single RX Avalon-MM master forwards all RX TLPs to the Qsys interconnect.

The Avalon-MM RX master module port has an 8-byte datapath in 64-bit mode and a 16-byte datapath in 128-bit mode. The Qsys interconnect fabric manages mismatched port widths transparently.

As Memory Request TLPs are received from the PCIe link, the most significant bits are used in the BAR matching as described in the PCI specifications. The least significant bits not used in the BAR match process are passed unchanged as the Avalon-MM address for that BAR's RX Master port.

For example, consider the following configuration specified using the Base Address Registers in the parameter editor:

1. BAR1:0 is a **64-bit prefetchable memory** that is **4KBytes -12 bits**
2. System software programs BAR1:0 to have a base address of 0x0000123456789000
3. A TLP received with address 0x0000123456789870
4. The upper 52 bits (0x0000123456789) are used in the BAR matching process, so this request matches.
5. The lower 12 bits, 0x870, are passed through as the Avalon address on the Rxm_BAR0 Avalon-MM Master port. The BAR matching software replaces the upper 20 bits of the address with the Avalon-MM base address.

Related Information

Avalon-MM-to-PCI Express Address Translation Algorithm for 32-Bit Addressing on page 9-17

Minimizing BAR Sizes and the PCIe Address Space

For designs that include multiple BARs, you may need to modify the base address assignments auto-assigned by Qsys in order to minimize the address space that the BARs consume. For example, consider a Qsys system with the following components:

- **Offchip_Data_Mem DDR3** (SDRAM Controller with UniPHY) controlling 256 MBytes of memory—Qsys auto-assigned a base address of 0x00000000
- **Quick_Data_Mem** (On-Chip Memory (RAM or ROM)) of 4 KBytes—Qsys auto-assigned a base address of 0x10000000
- **Instruction_Mem** (On-Chip Memory (RAM or ROM)) of 64 KBytes—Qsys auto-assigned a base address of 0x10020000
- **PCIe** (Avalon-MM Arria V Hard IP for PCI Express)
  - **Cra** (Avalon-MM Slave)—auto assigned base address of 0x10004000
  - **Rxm_BAR0** connects to **Offchip_Data_Mem DD R3 avl**
  - **Rxm_BAR2** connects to **Quick_Data_Mem s1**
  - **Rxm_BAR4** connects to PCIe. **Cra Avalon MM Slave**
- **Nios2** (Nios® II Processor)
  - **data_master** connects to **PCIe Cra**, **Offchip_Data_Mem DDR3 avl**, **Quick_Data_Mem s1**, **Instruction_Mem s1**, **Nios2 jtag_debug_module**
  - **instruction_master** connects to **Instruction_Mem s1**

Figure 9-6: Qsys System for PCI Express with Poor Address Space Utilization

The following figure uses a filter to hide the Conduit interfaces that are not relevant in this discussion.

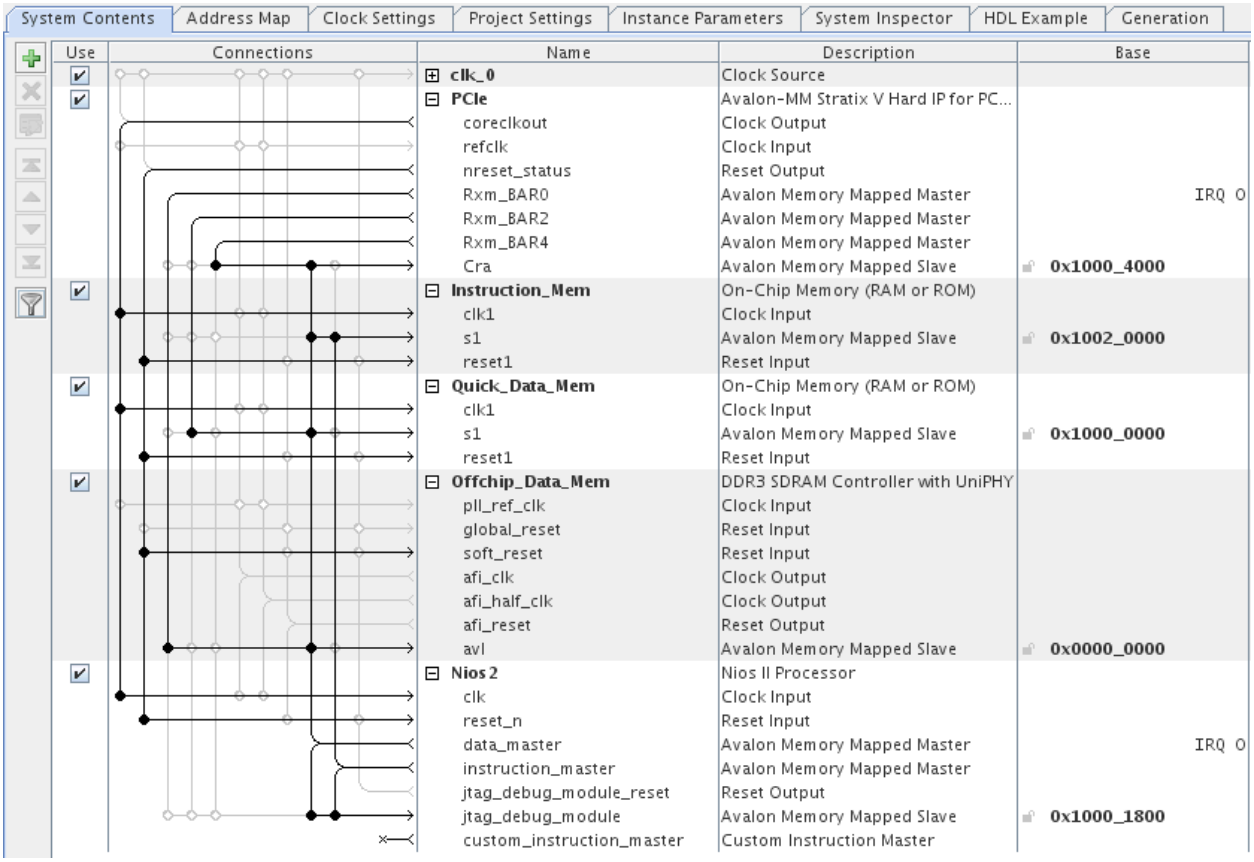
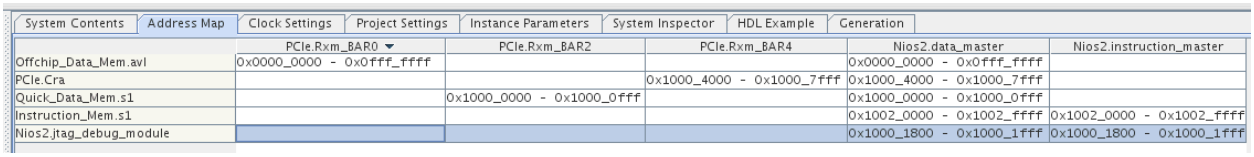


Figure 9-7: Poor Address Map

The following figure illustrates the address map for this system.



	PCIe.Rxm_BAR0	PCIe.Rxm_BAR2	PCIe.Rxm_BAR4	Nios2.data_master	Nios2.instruction_master
Offchip_Data_Mem.avl	0x0000_0000 - 0x0fff_ffff			0x0000_0000 - 0x0fff_ffff	
PCIe.Cra			0x1000_4000 - 0x1000_7fff	0x1000_4000 - 0x1000_7fff	
Quick_Data_Mem.s1		0x1000_0000 - 0x1000_0fff		0x1000_0000 - 0x1000_0fff	
Instruction_Mem.s1				0x1002_0000 - 0x1002_ffff	0x1002_0000 - 0x1002_ffff
Nios2.jtag_debug_module				0x1000_1800 - 0x1000_1fff	0x1000_1800 - 0x1000_1fff

The auto-assigned base addresses result in the following three large BARs:

- BAR0 is 28 bits. This is the optimal size because it addresses the **Offchip_Data_Mem** which requires 28 address bits.
- BAR2 is 29 bits. BAR2 addresses the **Quick_Data_Mem** which is 4 KBytes;. It should only require 12 address bits; however, it is consuming 512 MBytes of address space.
- BAR4 is also 29 bits. BAR4 address **PCIe Cra** is 16 KBytes. It should only require 14 address bits; however, it is also consuming 512 MBytes of address space.

This design is consuming 1.25 GB of PCIe address space when only 276 MBytes are actually required. The solution is to edit the address map to place the base address of each BAR at 0x0000_0000. The following figure illustrates the optimized address map.

Figure 9-8: Optimized Address Map

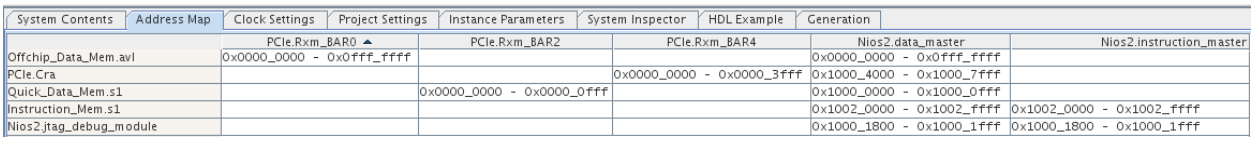
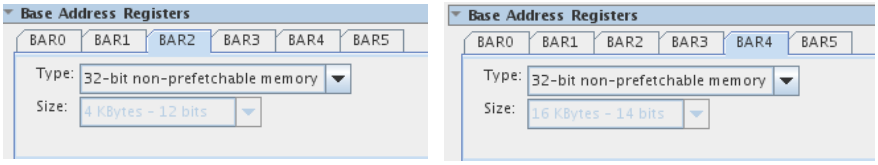


Figure 9-9: Reduced Address Bits for BAR2 and BAR4

The following figure shows the number of address bits required when the smaller memories accessed by BAR2 and BAR4 have a base address of 0x0000_0000.



For cases where the BAR Avalon-MM RX master port connects to more than one Avalon-MM slave, assign the base addresses of the slaves sequentially and place the slaves in the smallest power-of-two-sized address space possible. Doing so minimizes the system address space used by the BAR.

Related Information

Address Map Tab (Qsys)

Avalon-MM-to-PCI Express Address Translation Algorithm for 32-Bit Addressing

Note: The PCI Express-to-Avalon-MM bridge supports both 32- and 64-bit addresses. If you select 64-bit addressing the bridge does not perform address translation.

When you specify 32-bit addresses, the Avalon-MM address of a received request on the TX Avalon-MM slave port is translated to the PCI Express address before the request packet is sent to the Transaction Layer. You can specify up to 512 address pages and sizes ranging from 4 KByte to 4 GBytes when you customize your Avalon-MM Arria V Hard IP for PCI Express as described in *Avalon to PCIe Address Translation Settings* . This address translation process proceeds by replacing the MSB of the Avalon-MM address with the value from a specific translation table entry; the LSB remains unchanged. The number of MSBs to be replaced is calculated based on the total address space of the upstream PCI Express devices that the Avalon-MM Hard IP for PCI Express can access. The number of MSB bits is defined by the difference between the maximum number of bits required to represent the address space supported by the upstream PCI Express device minus the number of bits required to represent the **Size of address pages** which are the LSB pass-through bits (*N*). The **Size of address pages** (*N*) is applied to all entries in the translation table.

Each of the 512 possible entries corresponds to the base address of a PCI Express memory segment of a specific size. The segment size of each entry must be identical. The total size of all the memory segments is used to determine the number of address MSB to be replaced. In addition, each entry has a 2-bit field,

`Sp[1:0]`, that specifies 32-bit or 64-bit PCI Express addressing for the translated address. The most significant bits of the Avalon-MM address are used by the interconnect fabric to select the slave port and are not available to the slave. The next most significant bits of the Avalon-MM address index the address translation entry to be used for the translation process of MSB replacement.

For example, if the core is configured with an address translation table with the following attributes:

- **Number of Address Pages—16**
- **Size of Address Pages—1 MByte**
- **PCI Express Address Size—64 bits**

then the values in the following figure are:

- $N = 20$ (due to the 1 MByte page size)
- $Q = 16$ (number of pages)
- $M = 24$ (20 + 4 bit page selection)
- $P = 64$

In this case, the Avalon address is interpreted as follows:

- Bits [31:24] select the TX slave module port from among other slaves connected to the same master by the system interconnect fabric. The decode is based on the base addresses assigned in Qsys.
- Bits [23:20] select the address translation table entry.
- Bits [63:20] of the address translation table entry become PCI Express address bits [63:20].
- Bits [19:0] are passed through and become PCI Express address bits [19:0].

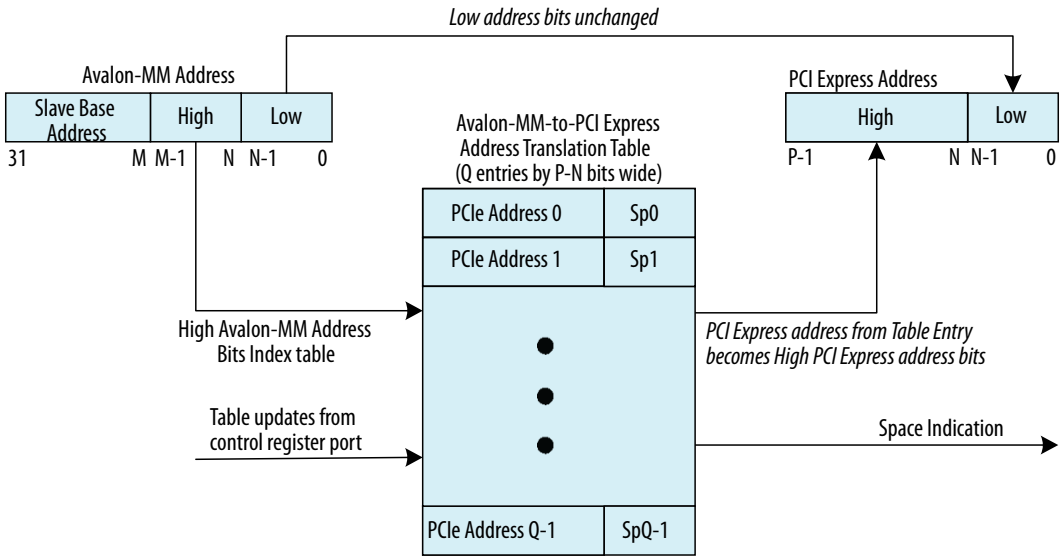
The address translation table is dynamically configured at run time. The address translation table is implemented in memory and can be accessed through the CRA slave module. Dynamic configuration is optimal in a typical PCI Express system where address allocation occurs after BIOS initialization.

For more information about how to access the dynamic address translation table through the CRA slave, refer to the “Avalon-MM-to-PCI Express Address Translation Table 0x1000–0x1FFF” on page 9–17.

Figure 9-10: Avalon-MM-to-PCI Express Address Translation

The following figure depicts the Avalon-MM-to-PCI Express address translation process. In this figure the variables represent the following paramers:

- *N*—the number of pass-through bits.
- *M*—the number of Avalon-MM address bits.
- *P*—the number of PCIe address bits.
- *Q*—the number of translation table entries.
- `Sp[1:0]`—the space indication for each entry.



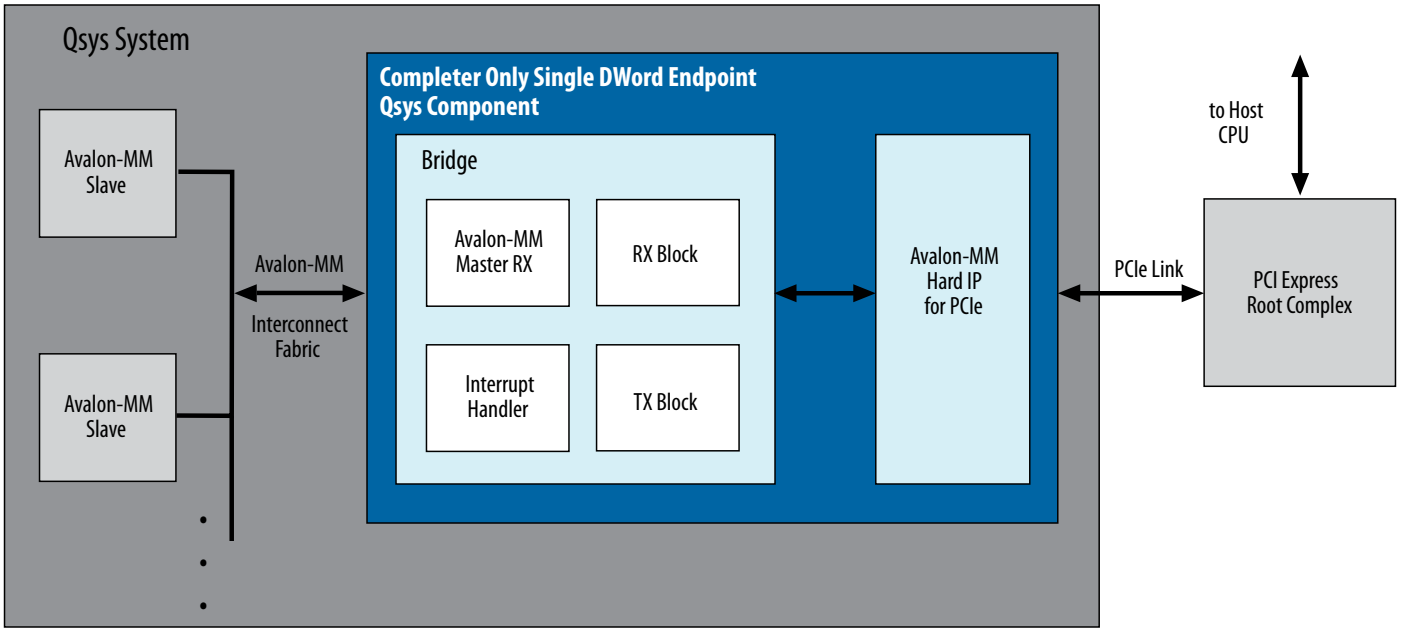
Completer Only Single Dword Endpoint

The completer only single dword endpoint is intended for applications that use the PCI Express protocol to perform simple read and write register accesses from a host CPU. The completer only single dword endpoint is a hard IP implementation available for Qsys systems, and includes an Avalon-MM interface to the Application Layer. The Avalon-MM interface connection in this variation is 32 bits wide. This endpoint is not pipelined; at any time a single request can be outstanding.

The completer-only single dword endpoint supports the following requests:

- Read and write requests of a single dword (32 bits) from the Root Complex
- Completion with Completer Abort status generation for other types of non-posted requests
- INTX or MSI support with one Avalon-MM interrupt source

Figure 9-11: Qsys Design Including Completer Only Single Dword Endpoint for PCI Express



The above figure shows the that completer-only single dword endpoint connects to a PCI Express root complex. A bridge component includes the Arria V Hard IP for PCI Express TX and RX blocks, an Avalon-MM RX master, and an interrupt handler. The bridge connects to the FPGA fabric using an Avalon-MM interface. The following sections provide an overview of each block in the bridge.

RX Block

The RX Block control logic interfaces to the hard IP block to process requests from the root complex. It supports memory reads and writes of a single dword. It generates a completion with Completer Abort (CA) status for read requests greater than four bytes and discards all write data without further action for write requests greater than four bytes.

The RX block passes header information to the Avalon-MM master, which generates the corresponding transaction to the Avalon-MM interface. The bridge accepts no additional requests while a request is being processed. While processing a read request, the RX block deasserts the `ready` signal until the TX block sends the corresponding completion packet to the hard IP block. While processing a write request, the RX block sends the request to the Avalon-MM interconnect fabric before accepting the next request.

Avalon-MM RX Master Block

The 32-bit Avalon-MM master connects to the Avalon-MM interconnect fabric. It drives read and write requests to the connected Avalon-MM slaves, performing the required address translation. The RX master supports all legal combinations of byte enables for both read and write requests.

For more information about legal combinations of byte enables, refer to *Avalon Memory Mapped Interfaces* in the Avalon Interface Specifications.

Related Information

Avalon Interface Specifications

TX Block

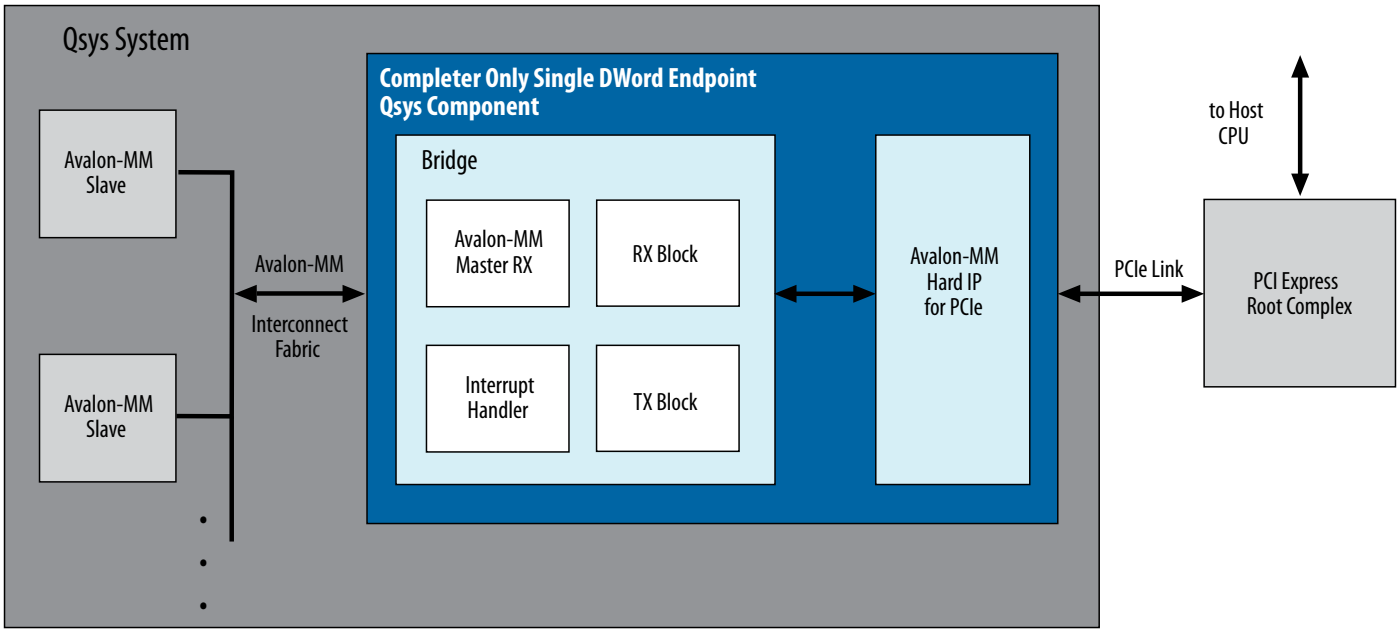
The TX block sends completion information to the Avalon-MM Hard IP for PCI Express which sends this information to the root complex. The TX completion block generates a completion packet with Completer Abort (CA) status and no completion data for unsupported requests. The TX completion block also supports the zero-length read (flush) command.

Interrupt Handler Block

The interrupt handler implements both INTX and MSI interrupts. The `msi_enable` bit in the configuration register specifies the interrupt type. The `msi_enable_bit` is part of the MSI message control portion in the MSI Capability structure. It is bit[16] of address 0x050 in the Configuration Space registers. If the `msi_en able` bit is on, an MSI request is sent to the Arria V Hard IP for PCI Express when received, otherwise INTX is signaled. The interrupt handler block supports a single interrupt source, so that software may assume the source. You can disable interrupts by leaving the interrupt signal unconnected in the IRQ column of Qsys.

When the MSI registers in the Configuration Space of the Completer Only Single Dword Arria V Hard IP for PCI Express are updated, there is a delay before this information is propagated to the Bridge module shown in the following figure.

Figure 9-12: Qsys Design Including Completer Only Single Dword Endpoint for PCI Express



You must allow time for the Bridge module to update the MSI register information. Normally, setting up MSI registers occurs during enumeration process. Under normal operation, initialization of the MSI registers should occur substantially before any interrupt is generated. However, failure to wait until the update completes may result in any of the following behaviors:

- Sending a legacy interrupt instead of an MSI interrupt
- Sending an MSI interrupt instead of a legacy interrupt
- Loss of an interrupt request

According to the *PCI Express Base Specification*, if `MSI_enable`=0 and the `Disable Legacy Interrupt bit`=1 in the Configuration Space `Command` register (0x004), the Hard IP should not send legacy interrupt messages when an interrupt is generated.

Design Implementation 10

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

Completing your design includes additional steps to specify analog properties, pin assignments, and timing constraints.

Making Analog QSF Assignments Using the Assignment Editor

You specify the analog parameters using the Quartus II Assignment Editor, the Pin Planner, or through the Quartus II Settings File **.(qsf)**.

Table 10-1: Power Supply Voltage Requirements

Data Rate	V_{CCR_GXB} and V_{CCT_GXB}	V_{CCA_GXB}
Arria V GX: Gen1 and Gen2	1.1 V	2.5 V
Arria V GT: Gen1 and Gen2	1.2 V	2.5 V

The Quartus II software provides default values for analog parameters. You can change the defaults using the Assignment Editor or the Pin Planner. You can also edit your **.qsf** directly or by typing commands in the Quartus II Tcl Console.

The following example shows how to change the value of the voltages required:

1. On the Assignments menu, select **Assignment Editor**. The Assignment Editor appears.
2. Complete the following steps for each pin requiring the V_{CCR_GXB} and V_{CCT_GXB} voltage:
   a. Double-click in the **Assignment Name** column and scroll to the bottom of the available assignments.
   b. Select **VCCR_GXB/VCCT_GXB Voltage**.
   c. In the **Value** column, select **1_1V** from the list.
3. Complete the following steps for each pin requiring the V_{CCA_GXB} voltage:
   a. Double-click in the **Assignment Name** column and scroll to the bottom of the available assignments.
   b. Select **VCCA_GXB Voltage**.
   c. In the **Value** column, select **3_0V** from the list.

The Quartus II software adds these instance assignments commands to the **.qsf** file for your project.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

You can also enter these commands at the Quartus II Tcl Console. For example, the following command sets the XCVR_VCCR_VCCT_VOLTAGE to 1.0 V for the pin specified:

```
set_instance_assignment -name XCVR_VCCR_VCCT_VOLTAGE 1_0V to “pin”
```

Related Information

- **Arria V GT, GX, ST, and SX Device Family Pin Connection Guidelines**
- **Arria V Device Datasheet**

Making Pin Assignments

Before running Quartus II compilation, use the **Pin Planner** to assign I/O standards to the pins of the device. Complete the following steps to bring up the **Pin Planner** and assign the 1.5-V pseudo-current mode logic (PCML) I/O standard to the serial data input and output pins:

1. On the Quartus II **Assignments** menu, select **Pin Planner**. The **Pin Planner** appears.
2. In the **Node Name** column, locate the PCIe serial data pins.
3. In the **I/O Standard** column, double-click the right-hand corner of the box to bring up a list of available I/O standards.
4. Select **1.5 V PCML I/O** standard.

Note: The IP core automatically assigns other required PMA analog settings, including 100 ohm internal termination.

Recommended Reset Sequence to Avoid Link Training Issues

1. Wait until the FPGA is configured as indicated by the assertion of CONFIG_DONE from the FPGA block controller.
2. Deassert the mgmt_rst_reset input to the Transceiver Reconfiguration Controller IP Core.
3. Wait for tx_cal_busy and rx_cal_busy SERDES outputs to be deasserted.
4. Deassert pin_perstn to take the Hard IP for PCIe out of reset. For plug-in cards, the minimum assertion time for pin_perstn is 100 ms. Embedded systems do not have a minimum assertion time for pin_perstn.
5. Wait for thereset_status output to be deasserted
6. Deassert the reset output to the Application Layer.

Related Information

Reset Sequence for Hard IP for PCI Express IP Core and Application Layer on page 6-3

Optional Features 11

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

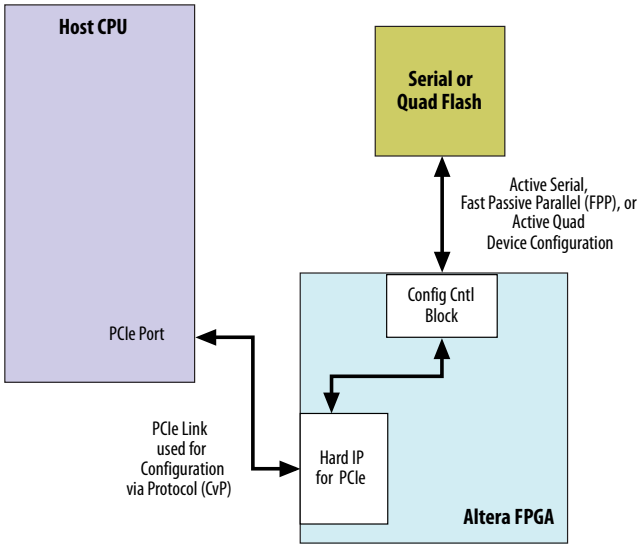
Configuration via Protocol (CvP)

The Hard IP for PCI Express architecture has an option to configure the FPGA and initialize the PCI Express link. In prior devices, a single Program Object File (**.pof**) programmed the I/O ring and FPGA fabric before the PCIe link training and enumeration began. The **.pof** file is divided into two parts:

- The I/O bitstream contains the data to program the I/O ring, the Hard IP for PCI Express, and other elements that are considered part of the periphery image.
- The core bitstream contains the data to program the FPGA fabric.

When you select the CvP design flow, the I/O ring and PCI Express link are programmed first, allowing the PCI Express link to reach the L0 state and begin operation independently, before the rest of the core is programmed. After the PCI Express link is established, it can be used to program the rest of the device. The following figure shows the blocks that implement CvP.

Figure 11-1: CvP in Arria V Devices


Host CPU
Serial or Quad Flash
Active Serial, Fast Passive Parallel (FPP), or Active Quad Device Configuration
Config Cntl Block
PCIe Port
PCIe Link used for Configuration via Protocol (CvP)
Hard IP for PCIe
Altera FPGA

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

CvP has the following advantages:

- Provides a simpler software model for configuration. A smart host can use the PCIe protocol and the application topology to initialize and update the FPGA fabric.
- Enables dynamic core updates without requiring a system power down.
- Improves security for the proprietary core bitstream.
- Reduces system costs by reducing the size of the flash device to store the **.pof**.
- Facilitates hardware acceleration.
- May reduce system size because a single CvP link can be used to configure multiple FPGAs.

Table 11-1: CvP Support

CvP is available for the following configurations.

Data Rate and Application Interface Width	Support
Gen1 128-bit interface to Application Layer	Supported
Gen2 128-bit interface to Application Layer	Contact your Altera sales representative

Note: You cannot use dynamic transceiver reconfiguration for the transceiver channels in the CvP-enabled Hard IP when CvP is enabled.

Related Information

Configuration via Protocol (CvP) Implementation in Altera FPGAs User Guide

ECRC

ECRC ensures end-to-end data integrity for systems that require high reliability. You can specify this option under the **Error Reporting** heading. The ECRC function includes the ability to check and generate ECRC. In addition, the ECRC function can forward the TLP with ECRC to the RX port of the Application Layer. When using ECRC forwarding mode, the ECRC check and generation are performed in the Application Layer.

You must turn on **Advanced error reporting (AER)**, **ECRC checking**, and **ECRC generation** under the **PCI Express/PCI Capabilities** heading using the parameter editor to enable this functionality.

For more information about error handling, refer to *Error Signaling and Logging* in Section 6.2 of the *PCI Express Base Specification*.

ECRC on the RX Path

When the **ECRC generation** option is turned on, errors are detected when receiving TLPs with a bad ECRC. If the **ECRC generation** option is turned off, no error detection occurs. If the **ECRC forwarding** option is turned on, the ECRC value is forwarded to the Application Layer with the TLP. If the **ECRC forwarding** option is turned off, the ECRC value is not forwarded.

Table 11-2: ECRC Operation on RX Path

ECRC Forwarding	ECRC Check Enable (5)	ECRC Status	Error	TLP Forward to Application Layer
No	No	none	No	Forwarded
		good	No	Forwarded without its ECRC
		bad	No	Forwarded without its ECRC
	Yes	none	No	Forwarded
		good	No	Forwarded without its ECRC
		bad	Yes	Not forwarded
Yes	No	none	No	Forwarded
		good	No	Forwarded with its ECRC
		bad	No	Forwarded with its ECRC
	Yes	none	No	Forwarded
		good	No	Forwarded with its ECRC
		bad	Yes	Not forwarded

ECRC on the TX Path

When the **ECRC generation** option is on, the TX path generates ECRC. If you turn on **ECRC forwarding**, the ECRC value is forwarded with the TLP. The following table summarizes the TX ECRC generation and forwarding. All unspecified cases are unsupported and the behavior of the Hard IP is unknown.In this table, if `TD` is 1, the TLP includes an ECRC. `TD` is the TL digest bit of the TL packet described in Appendix A, Transaction Layer Packet (TLP) Header Formats.

(5) The `ECRC Check Enable` field is in the `Configuration Space Advanced Error Capabilities and Control Register`.

Table 11-3: ECRC Generation and Forwarding on TX Path

All unspecified cases are unsupported and the behavior of the Hard IP is unknown.

ECRC Forwarding	ECRC Generation Enable [6]	TLP on Application	TLP on Link	Comments
No	No	TD=0, without ECRC	TD=0, without ECRC	
		TD=1, without ECRC	TD=0, without ECRC	
	Yes	TD=0, without ECRC	TD=1, with ECRC	ECRC is generated
		TD=1, without ECRC	TD=1, with ECRC	
Yes	No	TD=0, without ECRC	TD=0, without ECRC	Core forwards the ECRC
		TD=1, with ECRC	TD=1, with ECRC	
	Yes	TD=0, without ECRC	TD=0, without ECRC	
		TD=1, with ECRC	TD=1, with ECRC	

Related Information

Transaction Layer Packet (TLP) Header Formats on page 14-1

[6] The ECRC Generation Enable field is in the Configuration Space Advanced Error Capabilities and Control Register.

Transceiver PHY IP Reconfiguration 12

2014.12.15

UG-01105_avmm Subscribe Send Feedback

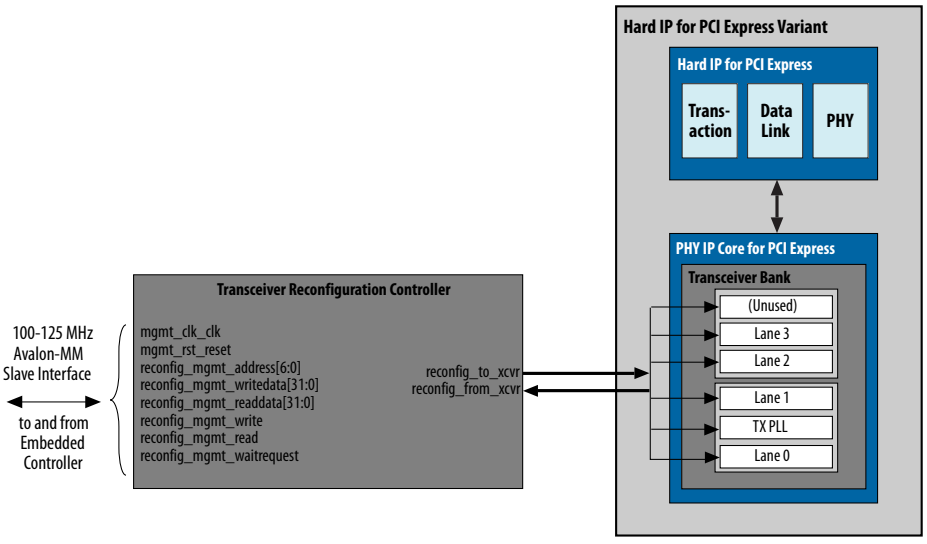
As silicon progresses towards smaller process nodes, circuit performance is affected by variations due to process, voltage, and temperature (PVT). Designs typically require offset cancellation to ensure correct operation. At Gen2 data rates, designs also require DCD calibration. Altera's Qsys example designs all include Transceiver Reconfiguration Controller and Altera PCIe Reconfig Driver IP cores to perform these functions.

Connecting the Transceiver Reconfiguration Controller IP Core

The Transceiver Reconfiguration Controller IP Core is available for V-series devices and can be found in the **Interface Protocols/Transceiver PHY** category in the IP Catalog. When you instantiate the Transceiver Reconfiguration Controller the **Enable offset cancellation block** and **Enable PLL calibration** options are enabled by default.

Figure 12-1: Altera Transceiver Reconfiguration Controller Connectivity

The following figure shows the connections between the Transceiver Reconfiguration Controller instance and the PHY IP Core for PCI Express instance for a ×4 variant.



© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

As this figure illustrates, the `reconfig_to_xcvr[ <n> 70-1:0]` and `reconfig_from_xcvr[ <n> 46-1:0]` buses connect the two components. You must provide a 100–125 MHz free-running clock to the `mgmt_clk_clk` clock input of the Transceiver Reconfiguration Controller IP Core.

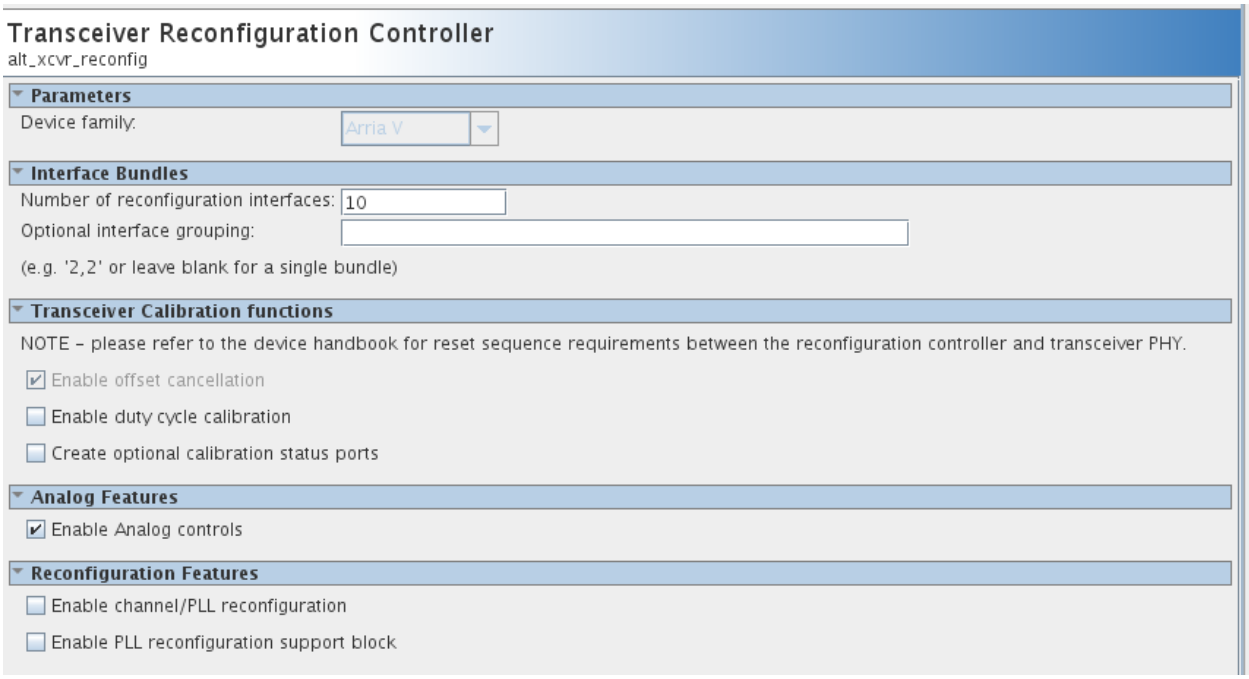
Initially, each lane and TX PLL require a separate reconfiguration interface. The parameter editor reports this number in the message pane. You must take note of this number so that you can enter it as a parameter value in the Transceiver Reconfiguration Controller parameter editor. The following figure illustrates the messages reported for a Gen2 ×4 variant. The variant requires five interfaces: one for each lane and one for the TX PLL.

Figure 12-2: Number of External Reconfiguration Controller Interfaces

ep_g2x4.DUT	5 reconfiguration interfaces are required for connection to the external reconfiguration controller and the reconfig driver
ep_g2x4.DUT	Credit allocation in the 16 KBytes receive buffer:
ep_g2x4.DUT	Posted : header=16 data=16
ep_g2x4.DUT	Non posted: header=16 data=0
ep_g2x4.DUT	Completion: header=195 data=781

When you instantiate the Transceiver Reconfiguration Controller, you must specify the required **Number of reconfiguration interfaces** as the following figure illustrates.

Figure 12-3: Specifying the Number of Transceiver Interfaces for Arria V and Cyclone V Devices



The Transceiver Reconfiguration Controller includes an **Optional interface grouping** parameter. Transceiver banks include six channels. For a ×4 variant, no special interface grouping is required because all 4 lanes and the TX PLL fit in one bank.

Note: Although you must initially create a separate logical reconfiguration interface for each lane and TX PLL in your design, when the Quartus II software compiles your design, it reduces the original

number of logical interfaces by merging them. Allowing the Quartus II software to merge reconfiguration interfaces gives the Fitter more flexibility in placing transceiver channels.

Note: You cannot use SignalTap to observe the reconfiguration interfaces.

Transceiver Reconfiguration Controller Connectivity for Designs Using CvP

If your design meets the following criteria:

- It enables CvP
- It includes an additional transceiver PHY that connect to the same Transceiver Reconfiguration Controller

then you must connect the PCIe `refclk` signal to the `mgmt_clk_clk` signal of the Transceiver Reconfiguration Controller and the additional transceiver PHY. In addition, if your design includes more than one Transceiver Reconfiguration Controller on the same side of the FPGA, they all must share the `mgmt_clk_clk` signal.

For more information about using the Transceiver Reconfiguration Controller, refer to the *Transceiver Reconfiguration Controller* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Related Information

Altera Transceiver PHY IP Core User Guide

Debugging 13

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

As you bring up your PCI Express system, you may face a number of issues related to FPGA configuration, link training, BIOS enumeration, data transfer, and so on. This chapter suggests some strategies to resolve the common issues that occur during hardware bring-up.

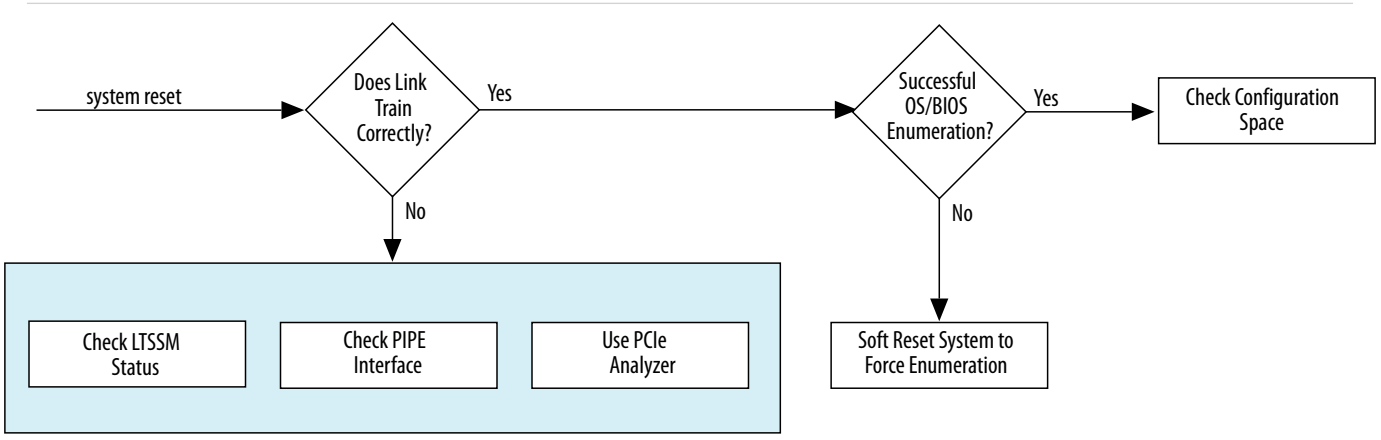
Hardware Bring-Up Issues

Typically, PCI Express hardware bring-up involves the following steps:

1. System reset
2. Link training
3. BIOS enumeration

The following sections, describe how to debug the hardware bring-up flow. Altera recommends a systematic approach to diagnosing bring-up issues as illustrated in the following figure.

Figure 13-1: Debugging Link Training Issues



Link Training

The Physical Layer automatically performs link training and initialization without software intervention. This is a well-defined process to configure and initialize the device's Physical Layer and link so that PCIe

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

packets can be transmitted. If you encounter link training issues, viewing the actual data in hardware should help you determine the root cause. You can use the following tools to provide hardware visibility:

- SignalTap II Embedded Logic Analyzer
- Third-party PCIe analyzer

You can use SignalTap II Embedded Logic Analyzer to diagnose the LTSSM state transitions that are occurring on the PIPE interface. The `ltssmstate[4:0]` bus encodes the status of LTSSM. The LTSSM state machine reflects the Physical Layer's progress through the link training process. For a complete description of the states these signals encode, refer to *Status, Link Training and Reset Signals*. When link training completes successfully and the link is up, the LTSSM should remain stable in the L0 state. When link issues occur, you can monitor `ltssmstate[4:0]` to determine the cause.

Related Information

Reset

Setting Up Simulation

Changing the simulation parameters reduces simulation time and provides greater visibility.

Changing Between Serial and PIPE Simulation

By default, the Altera testbench runs a serial simulation. You can change between serial and PIPE simulation by editing the top-level testbench file.

The `hip_ctrl_simu_mode_pipe` signal and `enable_pipe32_sim_hwtcl` parameter, specify serial or PIPE simulation. When both are set to 1'b0, the simulation runs in serial mode. When both are set to 1'b1, the simulation runs in PIPE mode. Complete the following steps to enable PIPE simulation. These steps assume that the actual testbench in Gen1 x4 with an Avalon-MM 64-bit interface.

1. In the top-level testbench, which is **<*working_dir*>/<*variant*>/testbench/<*variant*>_*tb*/simulation/<*variant*>_tb.v**, change the signal, `hip_ctrl_simu_mode_pipe` to 1'b1 as shown:

   ```
   pcie_de_gen1_x4_ast64 pcie_de_gen1_x4_ast64_x_inst (.dut_hip_ctrl_simu_mode_pipe
   ( 1'b1 ),
   ```

2. In the top-level HDL module for the Hard IP which is **<*working_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation/submodules/<*variant*>.v** change the parameter `enable_pipe32_sim_hwtcl` parameter to 1'b1 as shown:

   ```
   altpcie_<dev>_hip_ast_hwtcl #( .enable_pipe32_sim_hwtcl ( 1 ),
   ```

Using the PIPE Interface for Gen1 and Gen2 Variants

Running the simulation in PIPE mode reduces simulation time and provides greater visibility.

Complete the following steps to simulate using the PIPE interface:

1. Change to your simulation directory, **<*work_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation**
2. Open **<*variant*>_tb.v**.
3. Search for the string, `serial_sim_hwtcl`. Set the value of this parameter to 0 if it is 1.
4. Save **<*variant*>_tb.v**.

Reducing Counter Values for Serial Simulations

You can accelerate simulation by reducing the value of counters whose default values are set for hardware, not simulation.

Complete the following steps to reduce counter values for simulation:

1. Open **<*work_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation/submodules/altpcie_tbed_<*dev*>_hwtcl.v** .
2. Search for the string, `test_in`.
3. To reduce the value of several counters, set `test_in[0] = 1`.
4. Save **altpcietb_bfm_top_rp.v**.

Disable the Scrambler for Gen1 and Gen2 Simulations

The encoding scheme implemented by the scrambler applies a binary polynomial to the data stream to ensure enough data transitions between 0 and 1 to prevent clock drift. The data is decoded at the other end of the link by running the inverse polynomial.

Complete the following steps to disable the scrambler:

1. Open **<*work_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation/submodules/altpcie_tbed_<*dev*>_hwtcl.v**.
2. Search for the string, `test_in`.
3. To disable the scrambler, set `test_in[2] = 1`.
4. Save **altpcie_tbed_sv_hwtcl.v**.

Changing between the Hard and Soft Reset Controller

The Hard IP for PCI Express includes both hard and soft reset control logic. By default, Gen1 devices use the Hard Reset Controller. Gen2 devices use the soft reset controller. For variants that use the hard reset controller, changing to the soft reset controller provides greater visibility.

Complete the following steps to change to the soft reset controller:

1. Open **<*work_dir*>/<*variant*>/testbench/<*variant*>_tb/simulation/submodules/<*variant*>.v**.
2. Search for the string, `hip_hard_reset_hwtcl`.
3. If `hip_hard_reset_hwtcl = 1`, the hard reset controller is active. Set `hip_hard_reset_hwtcl = 0` to change to the soft reset controller.
4. Save **variant.v**.

Use Third-Party PCIe Analyzer

A third-party logic analyzer for PCI Express records the traffic on the physical link and decodes traffic, saving you the trouble of translating the symbols yourself. A third-party logic analyzer can show the two-way traffic at different levels for different requirements. For high-level diagnostics, the analyzer shows the LTSSM flows for devices on both side of the link side-by-side. This display can help you see the link training handshake behavior and identify where the traffic gets stuck. A traffic analyzer can display the contents of packets so that you can verify the contents. For complete details, refer to the third-party documentation.

BIOS Enumeration Issues

Both FPGA programming (configuration) and the initialization of a PCIe link require time. Potentially, an Altera FPGA including a Hard IP block for PCI Express may not be ready when the OS/BIOS begins enumeration of the device tree. If the FPGA is not fully programmed when the OS/BIOS begins its enumeration, the OS does not include the Hard IP for PCI Express in its device map.

You can use either of the following two methods to eliminate this issue:

- You can perform a soft reset of the system to retain the FPGA programming while forcing the OS/BIOS to repeat its enumeration.
- You can use CvP to program the device.

Transaction Layer Packet (TLP) Header Formats

A

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

The following figures show the header format for TLPs without a data payload.

Figure A-1: Memory Read Request, 32-Bit Addressing

Memory Read Request, 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure A-2: Memory Read Request, Locked 32-Bit Addressing

Memory Read Request, Locked 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure A-3: Memory Read Request, 64-Bit Addressing

Memory Read Request, 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Figure A-4: Memory Read Request, Locked 64-Bit Addressing

Memory Read Request, Locked 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	0	0	0	0	1	0	TC			0	0	0	0	T	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Figure A-5: Configuration Read Request Root Port (Type 1)

Configuration Read Request Root Port (Type 1)

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device No					Func			0	0	0	0	Ext Reg				Register No						0	0
Byte 12	Reserved																															

Figure A-6: I/O Read Request

I/O Read Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure A-7: Message without Data

Message without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	1	1	0	r 2	r 1	r 0	0	TC			0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros																															
Byte 12	Vendor defined or all zeros																															

Note

(1) Not supported in Avalon-MM.

Figure A-8: Completion without Data

Completion without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure A-9: Completion Locked without Data

Completion Locked without Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

TLP Packet Formats with Data Payload

Figure A-10: Memory Write Request, 32-Bit Addressing

Memory Write Request, 32-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure A-11: Memory Write Request, 64-Bit Addressing

Memory Write Request, 64-Bit Addressing

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Requester ID																Tag								Last BE				First BE			
Byte 8	Address[63:32]																															
Byte 12	Address[31:2]																														0	0

Figure A-12: Configuration Write Request Root Port (Type 1)

Configuration Write Request Root Port (Type 1)

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Bus Number								Device No								0	0	0	0	Ext Reg				Register No						0	0
Byte 12	Reserved																															

Figure A-13: I/O Write Request

I/O Write Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	TD	EP	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Byte 4	Requester ID																Tag								0	0	0	0	First BE			
Byte 8	Address[31:2]																														0	0
Byte 12	Reserved																															

Figure A-14: Completion with Data

Completion with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Completer ID																Status			B	Byte Count											
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure A-15: Completion Locked with Data

Completion Locked with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	0	0	1	0	1	1	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Byte 4	Completer ID																Status				B	Byte Count										
Byte 8	Requester ID																Tag								0	Lower Address						
Byte 12	Reserved																															

Figure A-16: Message with Data

Message with Data

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0	0	1	1	1	0	r2	r1	r0	0	TC			0	0	0	0	TD	EP	0	0	0	0	Length									
Byte 4	Requester ID																Tag								Message Code							
Byte 8	Vendor defined or all zeros for Slot Power Limit																															
Byte 12	Vendor defined or all zeros for Slots Power Limit																															

Lane Initialization and Reversal

B

2014.12.15

UG-01105_avmm

Subscribe Send Feedback

Connected components that include IP blocks for PCI Express need not support the same number of lanes. The ×4 variations support initialization and operation with components that have 1, 2, or 4 lanes. The ×8 variant supports initialization and operation with components that have 1, 2, 4, or 8 lanes.

Lane reversal permits the logical reversal of lane numbers for the ×1, ×2, ×4, and ×8 configurations. Lane reversal allows more flexibility in board layout, reducing the number of signals that must cross over each other when routing the PCB.

Table B-1: Lane Assignments without Lane Reversal

Lane Number	7	6	5	4	3	2	1	0
×8 IP core	7	6	5	4	3	2	1	0
×4 IP core	—	—	—	—	3	2	1	0
×1 IP core	—	—	—	—	—	—	—	0

Table B-2: Lane Assignments with Lane Reversal

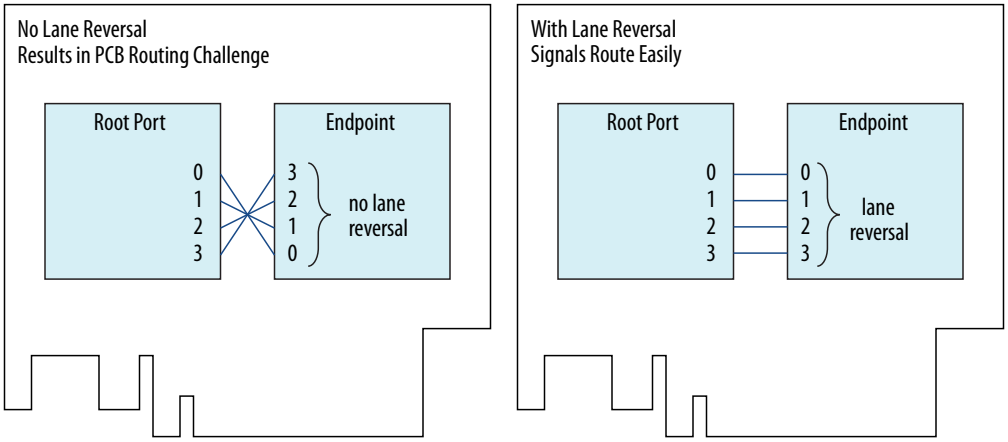
Core Config	8				4				1			
Slot Size	8	4	2	1	8	4	2	1	8	4	2	1
Lane pairings	7:0,6:1,5:2, 4:3, 3:4,2:5, 1:6,0:7	3:4,2:5, 1:6,0:7	1:6, 0:7	0:7	7:0,6:1, 5:2,4:3	3:0,2:1, 1:2,0:3	3:0, 2:1	3:0	7:0	3:0	1:0	0:0

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure B-1: Using Lane Reversal to Solve PCB Routing Problems

The following figure illustrates a PCI Express card with ×4 IP Root Port and a ×4 Endpoint on the top side of the PCB. Connecting the lanes without lane reversal creates routing problems. Using lane reversal solves the problem.



Additional Information

C

2014.08.18

UG-01105_avmm

Subscribe

Send Feedback

Revision History for the Avalon-MM Interface

Date	Version	Changes Made
2014.12.15	14.1	Made the following changes to the user guide: • In the figured titled *Specifying the Number of Transceiver Interfaces for Arria V and Cyclone V Devices*, removed the **Calibrate duty cycle during power up**. Duty cycle calibration occurs during Gen1 to Gen2 speed changes. This is no longer a parameter than you can turn on and off. • Corrected discussion of soft and hard reset controllers. The hardened reset controller is used for Arria V and Cyclone V devices. • Added statement that the bottom left hard IP block includes the CvP functionality for flip chip packages. For other package types, the CvP functionality is in the bottom right block. • Corrected bit definitions for `CvP Status` register. • Updated definition of `CVP_NUMCLKS` in the `CvP Mode Control` register. • Added definitions for `test_in[2]`, `test_in[6]` and `test_in[7]`. • Removed requirement that `TxsWrite_i` be asserted continuously throughout a write burst. `TxsWrite_i` may be deasserted and reasserted during a burst. • Corrected *Channel Utilization* table for x1 instances. Data is driven on Channel 0. The CMU clock is on Channel 1.

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Date	Version	Changes Made
		• Added figure showing connectivity for the Transceiver Reconfiguration Controller and Altera PCIe Reconfig Driver IP Cores to the *Gettting Started* chapter. • Removed **Maximum** and **High** settings from the **RX Buffer credit allocation -performance for received requests** setting. These settings are not available for the Avalon-MM interface and could lead to data corruption. • Revised *Receiving a Completion TLP* under *Programming Model for Avalon-MM Root Port* to cover read and non-posted completions.
2014.06.30	14.0	Added the following features to the Arria V Avalon-MM Hard IP for PCI Express: • Added access to selected Configuration Space registers and link status registers through the optional Control Register Access (CRA) Avalon-MM slave port. • Added optional hard IP status bus that includes signals necessary to connect the Transceiver Reconfiguration Controller IP Core. • Added optional hard IP status extension bus which includes signals that are useful for debugging, including: link training, status, error, and Configuration Space signals. • For `TxsByteEnable_i[<w>-1:0]`, added restrictions on the legal patterns of enabled and disabled bytes. • Clarified the behavior of the `TxsWaitrequest_o` signal. Made the following changes to the user guide: • Created separate user guides for variants using the Avalon-MM, Avalon-ST, and Avalon-MM with DMA interfaces to the Application Layer. • Corrected frequency range for `hip_reconfig_clk`. It should be 100-125 MHz. • Simplified the *Getting Started* chapter. It copies the Gen1 x4 example from the install directory and does not include step-by-step instructions to recreate the design. • Added *Next Steps in Creating a Design for PCI Express* to *Datasheet* chapter. • Removed references to the MegaWizard® Plug-In Manager. In 14.0 the IP Parameter Editor Powered by Qsys has replaced the MegaWizard Plug-In Manager. • Added definition for `test_in[6]` and link to Knowledge Base Solution on observing the PIPE interface signals on the `test_out` bus.

Date	Version	Changes Made
		• Clarified that the Avalon-MM Bridge does not generate out-of-order Avalon-MM-to-PCI Express Read Completions even to different BARs. • Removed `reconfig_busy` port from connect between PHY IP Core for PCI Express and the Transceiver Reconfiguration Controller in the *Altera Transceiver Reconfiguration Controller Connectivity* figure. The Transceiver Reconfiguration Controller drives `reconfig_busy` port to the Altera PCIe Reconfig Driver. • Added fact that DCD calibration is required for Gen2 data rate in the description of the transceiver reconfiguration signals. Updated figure showing Transceiver Reconfiguration Controller parameter editor. • Removed reference to Gen2 x1 62.5 MHz configuration in *Application Layer Clock Frequency for All Combination of Link Width, Data Rate and Application Layer Interface Widths* table. This configuration is not supported. • Added description of `TxsWaitRequest` signal which is asserted when the Avalon-MM bridge has eight outstanding read requests. • Added sections on making analog QSF and pin assignments. • Enhanced definition of Device ID and Sub-system Vendor ID to say that these registers are only valid in the Type 0 (Endpoint) Configuration Space. • Improved figure showing multiple MSI and MSI-X support and added reference to example on Altera wiki. • Removed references to the ATX PLL. This PLL is not available for Arria V • Updated *Power Supply Voltage Requirements* table. • For Cyclone V devices changed speed grade recommendation to use GT devices for both the Gen1 and Gen2 data rate.
2014.12.20	13.1	Made the following changes: • Added constraints for `refclk` when CvP is enabled. • Corrected location information for `nPERSTL*`. • Corrected definition of `test_in[4:1]`. • In *Debugging* chapter, under changing between soft and hard reset controller, changed the file name in which the parameter `hip_hard_reset_hwtcl`must be set to 0 to use the soft reset controller. • Added explanation of channel labeling for serial data. The Hard IP on the left side of the device must connect to the appropriate channels on the left side of the device, and so on. • Corrected connection for the Transceiver Reconfiguration Controller IP Core reset signal, `alt_xcvr_reconfig_0 mgmt_rst_reset`, *Getting Started with the Avalon-MM Arria V Hard IP for PCI Express*. This reset input connects to `clk_0 clk_reset`.

Date	Version	Changes Made
		• Added definition of `nreset_status` for variants using the Avalon-MM interface. • In *Transaction Layer Routing Rules and Programming Model for Avalon-MM Root Port* , added the fact that Type 0 Configuration Requests sent to the Root Port are not filtered by the device number. Application Layer software must filter out requests for device number greater than 0. • Added *Recommended Reset Sequence to Avoid Link Training Issues* to the *Debugging* chapter. • Added limitation for `RxmIrq_<n>_i[<m>:0]` when interrupts are received on consecutive cycles. • Updated timing diagram for `tl_cfg_ctl`. • Removed I/O Read Request and I/O Write Requests from TLPs supported for Avalon-MM interface. • Added note that the LTSSM interface can be used for SignalTap debugging. • Added restriction on the use of dynamic transceiver reconfiguration when CvP is enabled.
2014.05.06	13.0	Made the following changes: • Timing models are now final. • Added instructions for running the Single Dword variant. • Corrected definition of `test_in[4:1]`. This vector must be set to 4'b0100. • Corrected connection for `mgmt_clk_clk` in Figure 3-2. • Corrected definition of `nPERSTL*`. The device has 1 `nPERSTL*` pin for each instance of the Hard IP for PCI Express in the device. • Corrected feature comparison table in *Datasheet* chapter. The Avalon-MM Hard IP for PCI Express IP Core does not support legacy endpoints.

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

Contact (1)	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com

Contact (1)	Contact Method	Address
Product literature	Website	www.altera.com/literature
Nontechnical support (general)	Email	nacomp@altera.com
(software licensing)	Email	authorization@altera.com

Note to Table:

1. You can also contact your local Altera sales office or sales representative.

Related Information

- **Technical Support**
- **Technical Training**
- **Customer Training**
- **Product Documentation**
- **Non-Technical Suport (general)**
- **Licensing**

Typographic Conventions

The following table shows the typographic conventions this document uses.

Table C-1: Visual CueMeaning

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, \ **qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, <*file name*> and <*project name*> **.pof** file.

Visual Cue	Meaning
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, "Typographic Conventions."
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
r	An angled arrow instructs you to press the Enter key.
1., 2., 3., anda., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets indicate a list of items when the sequence of the items is not important.
1	The hand points to information that requires special attention.
h	The question mark directs you to a software help system with related information.
f	The feet direct you to another document or website with related information.
m	The multimedia icon directs you to a related multimedia presentation.
c	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.

Visual Cue	Meaning
w	A warning calls attention to a condition or possible situation that can cause you injury.
The **Subscribe** button links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.	
The **Feedback** icon allows you to submit feedback to Altera about the document. Methods for collecting feedback vary as appropriate for each document.	

Related Information

Email Subscription Management Center